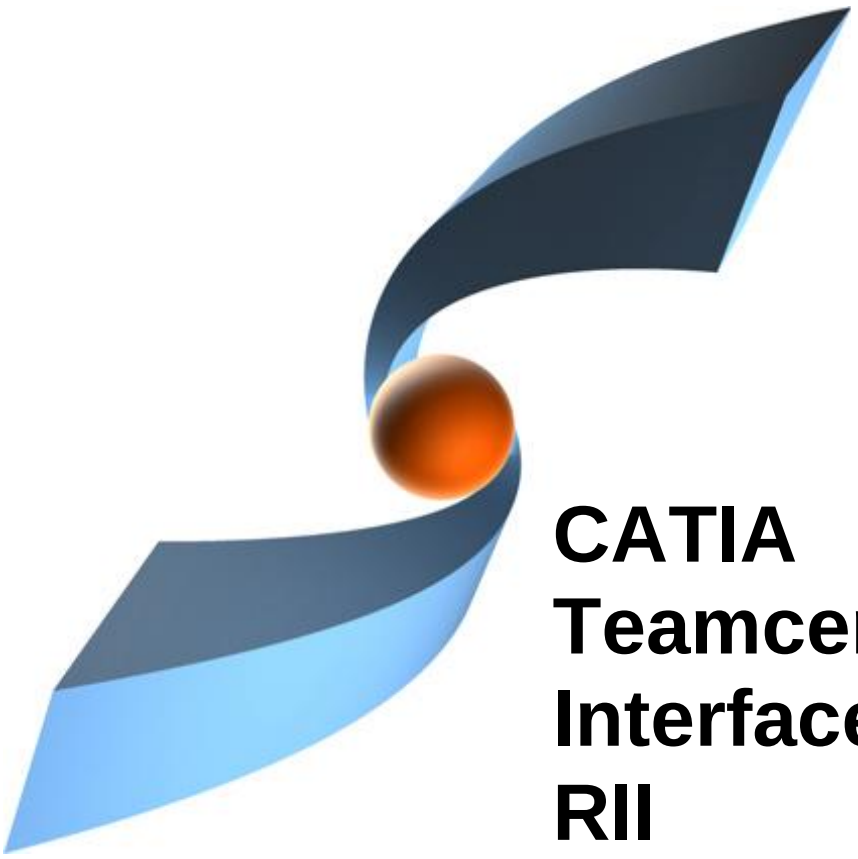

CATIA Teamcenter Interface RII

CMI RII Release 3.8

Customizing Manual

Copyright

© 2000, 2010, 2015 T-Systems International GmbH.

All rights reserved. Printed in Germany.

Contact

T-Systems International GmbH
GDC Product Lifecycle Management
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

http://www.cmi-support.com

☎ +49 (0)40 30600-5544
✉ +49 (0)391 5801-25688
mail : *cmi_support@t-systems.com*

Manual History

Version	Date
2.3	November 2010
3.0	April 2011
3.1	November 2011
3.2	April 2012
3.3	November 2012
3.4	April 2013
3.5	November 2013
3.6	April 2014
3.7	November 2014
3.8	April 2015

This edition obsoletes all previous editions.

Trademarks

CATIA is a registered trademark of Dassault Systèmes.

Teamcenter is a registered trademark of Siemens PLM Corporation.

Metaphase is a registered trademark of Metaphase Technology, Inc.

Names of other products mentioned in this manual are used only for identification purpose and may be trademarks of their companies.

Preface

About this Guide

This guide describes customizing and advanced configuration information for the CATIA Teamcenter Interface. Before using this guide, be sure you understand:

- the administration of the CATIA system
- the administration of the Teamcenter system
- the Java programming language
- the Eclipse RCP Development Environment

Related Documents

The following manuals contain information about installation, usage and customizing of CATIA Metaphase Interface:

Manual Title	Version
CATIA Teamcenter Interface RII Installation Manual	3.8
CATIA Teamcenter Interface RII User Manual	3.8

Your Comments are Welcome

Please feel free to tell us your opinion; we are always interested in improving our publications. Mail your comments to:

T-Systems International GmbH
GDC Product Lifecycle Management
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

E-mail: *cmi_support@t-systems.com*

Table of Contents

CHAPTER 1 ..... **1**

OVERVIEW **1**

INTRODUCING CMI (CATIA TEAMCENTER INTERFACE) 1
CMI RII JOINS THE ADVANTAGES OF CAD WITH PDM 1

CHAPTER 2 ..... **3**

SYSTEM ARCHITECTURE **3**

CHAPTER 3 ..... **5**

ASSEMBLY STRUCTURE **5**

PREDEFINED OBJECT STRUCTURE 5
ITEMREVISION RELATION OBJECTS 6

CHAPTER 4 ..... **7**

CUSTOMIZATION OPTIONS **7**

PREPARING CUSTOMIZATION TASKS 7
SETTING UP CATIA ENVIRONMENT 8
USED DATA TYPES FOR CUSTOMIZATION 9
SHOW TEAMCENTER META DATA IN CATIA 9
Customization: Show Teamcenter meta data in CATIA V5 9
STANDARD PROPERTIES IN CATIA V5 11
Customization: Sending standard attributes to CATIA V5 11
Customization: Receiving user defined attributes from CATIA V5 12
USER DEFINED PROPERTIES IN CATIA V5 14
Customization: Sending user defined attributes to CATIA V5 15
Customization: Receiving user defined attributes from CATIA V5 16
CONFIGURABLE BEHAVIORS IN CATIA V5 16
Descriptions of the behaviors: 17
CONFIGURABLE CHECK IN/OUT DIALOG IN CATIA V5 18
Customization: Receiving additional column attributes from CATIA V5 19
WORKING WITH CATIA V5 RELEASED CACHE 19
DESIGN TABLE SUPPORT 21
Configuration 21
Customizable extension points 22
MML SUPPORT 24
Installation 24
Configuration 24
Customizable extension points 25
TRANSFER MODEL INFOS TO TEAMCENTER 26
Weight properties (inertia) 26
Configuration 27
CATIA V5 Version information 27
Customizable extension points 27
SET BOM TYPE OF NEW CATIA FILES BY TEAMCENTER CUSTOMIZATION 29
VALIDATION OF CMI ARCHIVE NAMES 30
VALIDATION BEFORE RUNNING THE CATIA SYNCHRONIZE COMMAND 30
CUSTOMIZATION OPTION FOR CREATE 34
CUSTOMIZATION OPTION FOR READ 36
CATPROCESS CUSTOMIZATION 37
Customizable extension points 37
CATIA VERSION CHECK CUSTOMIZATION 38
MAPPING FILE CUSTOMIZATION 39
CUSTOMIZABLE NAMING SCHEMES FOR EXPORT 40
SCRIPT FILE CUSTOMIZATION 43

PRODUCT BOUNDING BOXES 43
Configuration *44*
DISPLAY CATIA NODE NAME IN SYNCHRONIZE 44
MODEL TYPE CUSTOMIZATION 46
ITEM REVISE CUSTOMIZATION 47
CAA CUSTOMIZATION 48
Configuration *49*

CHAPTER 5 51

TEAMCENTER CONFIGURATION VARIABLES 51

CMI TEAMCENTER PREFERENCES 51

CHAPTER 6 61

CMI CATIA V5 RII INSTALLATION PACKAGE STRUCTURE 61

DIRECTORIES 61
FILES 62
CUSTOMER DEPENDENT CONFIGURATIONS FOR CATIA V5 65
Environment settings *65*

Table of Figures

FIGURE 1: SYSTEM ARCHITECTURE OF CMI 3
FIGURE 2: PREDEFINED OBJECT STRUCTURE 5
FIGURE 3: USER DEFINED PROPERTIES 15
FIGURE 4: CHECK IN/CHECK OUT DIALOG 19
FIGURE 5: TEAMCENTER CMI RII OPTIONS 20
FIGURE 6: CATIA CACHE MANAGEMENT OPTIONS 20
FIGURE 7: CATIA *MASS* PROPERTIES 27
FIGURE 8: NAMING SCHEMA DIALOG 41
FIGURE 9: CMI SYNCHRONIZE TEAMCENTER DIALOG 45
FIGURE 10: CATIA NODE CUSTOMIZATION OPTIONS 45
FIGURE 11: DIRECTORY STRUCTURE OF THE CMI CATIA V5 RII INSTALLATION PACKAGE 61
FIGURE 12: DIRECTORY STRUCTURE OF THE CMICATV5 INSTALLATION DIRECTORY ON WINDOWS 32-BIT 62
FIGURE 13: EXAMPLE OF DIRECTORY STRUCTURE OF THE CMICATV5 INSTALLATION SUBDIRECTORY MSGCATALOG 65

CHAPTER 1

Overview

This chapter provides basic information about the *CATIA Teamcenter Interface* and lists some features of this application interface.

Introducing CMI (CATIA Teamcenter Interface)

The *CATIA Teamcenter Interface* (CMI RII) was developed by T-Systems as a high-end integration between the CAD system *CATIA* V5 and the PDM system Teamcenter. With this interface it is possible to manage *CATIA V4* and *CATIA V5* models and assemblies in *Teamcenter* and *CATIA V5*.

CATIA V5 uses assemblies similar to *Teamcenter*. CMI RII makes a bidirectional mapping between the *Teamcenter* structure and the *CATIA V5* structure. So users have the full functionality of *Teamcenter* and *CATIA V5*.

CMI RII joins the advantages of CAD with PDM

The *CATIA Teamcenter Interface* combines the CAD Excellency of CATIA with the power of the PDM system Teamcenter. It provides the user with a more sophisticated way of working with CATIA by allowing the management of product structures and multiple level assembly structures within the PDM system.

The CATIA Teamcenter Interface (CMI RII) permits:

- Integration of CATIA data in workflow (e.g. release control);
- Management of CATIA data in vaults, without knowledge about the underlying file system;
- Updating concurrent engineering processes by different users;
- Distribution of CATIA data in a network;
- Simultaneous management of CATIA data and structures;
- Construction of part structures within Teamcenter;
- Modification of the position of the structures;
- Search for CATIA data by different attributes.

CHAPTER 2

System Architecture

The components of CMI RII are:

CMI RII plugin	The Teamcenter Rich Application Client plugin to enhance the Rich Client with CMI RII functionality.
CMI RII Add-In	The CATIA V5 module to enhance the CATIA V5 with CMI RII functionality.
Exchange Map	A dedicated user directory on the client workstation. The CATIA extension expects the model files to be only within this directory.
CMI RII Server Library	The Teamcenter Server Library to customize Teamcenter for CMI RII

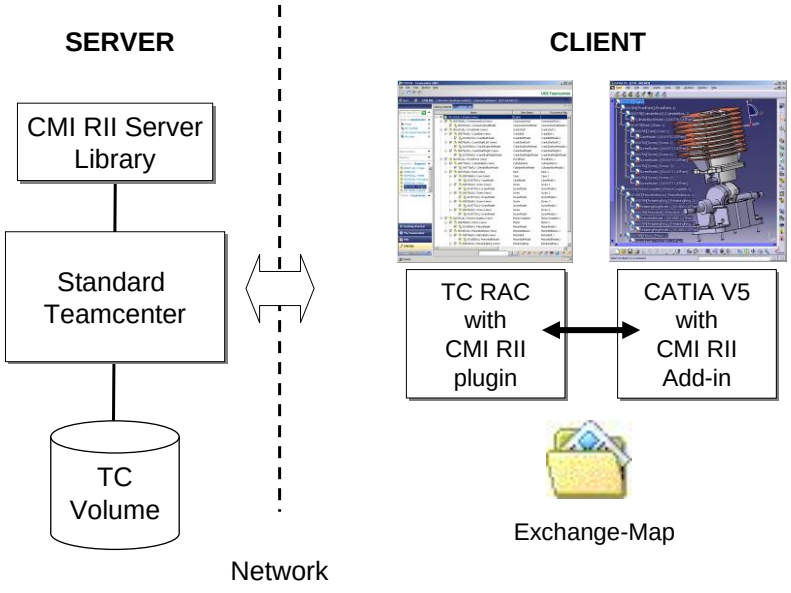

Figure 1: System architecture of CMI

The interaction can either start in CATIA or in Teamcenter.

CHAPTER 3

Assembly Structure

Predefined Object Structure

CMI RII uses the following pre-defined object structure in Teamcenter:

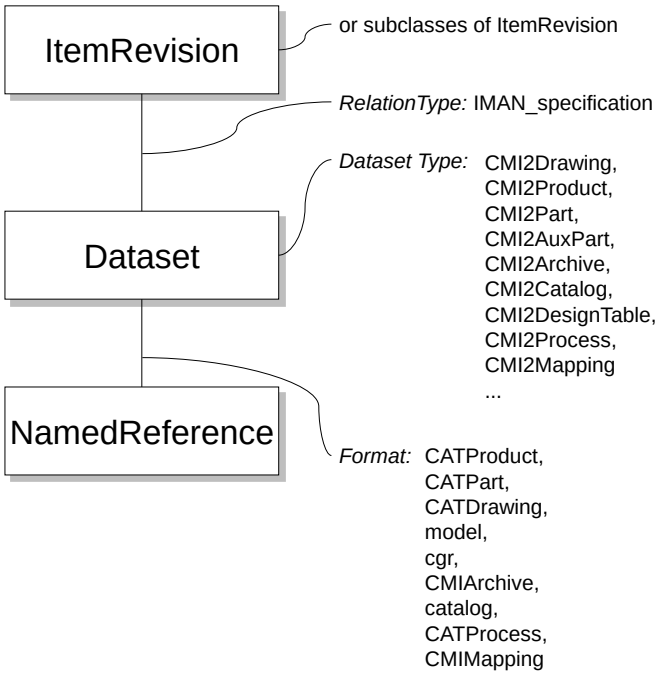

Figure 2: Predefined object structure

The default dataset types are:

CMI2Drawing	Dataset for drawings (CATDrawing).
CMI2Cgm	Dataset for cgms.
CMI2Product	Dataset for structure files (CATProduct).
CMI2Part	Dataset for BOM CATPart files
CMI2Model	Dataset for BOM model files.
CMI2Cgr	Dataset for BOM cgr files.
CMI2Rep	Dataset for BOM represntation files.
CMI2AuxPart	Dataset for auxiliary (non BOM) CATPart files.
CMI2AuxModel	Dataset for auxiliary (non BOM) model files.
CMI2AuxCgr	Dataset for auxiliary (non BOM) cgr files.
CMI2AuxRep	Dataset for auxiliary (non BOM) representation files.

CMI2Archive	Dataset for CMI Archive, non BOM substructures, also containing CATAnalysis, Electrical etc.
CMI2AuxAnalysis	Dataset for CATAnalysis files.
CMI2Catalog	Dataset for CATIA catalog files (catalog).
CMI2DesignTable	Dataset for Design Tables (txt, xls MS Excel).
CMI2Process	Dataset for process files (CATProcess).
CMI2Mapping	Dataset for mapping files (CMIMapping).
CMI2Script	Dataset for CATIA script files (CATScript).
CMI2CacheCgr	Dataset for cache cgr files used for global caching (cgr).
CMI2DerivedModel	Dataset for V4 model files used for the CMI RII V4 integration (model).

Each item revision object can have at least one CMI2Product or one CMI2Part (or CMI2Model, CMI2Cgr, or CMI2Rep).

An Item with a CMI2Product Dataset can have 0..n additional Datasets of type CMI2AuxPart, CMI2AuxModel, CMI2AuxCgr, CMI2AuxRep, and/or CMI2Archive.

ItemRevision Relation Objects

CMI RII uses the standard relations for the ItemRevision – ItemRevision relation.

The CATIA instance name is stored in the OccurenceName attribute of the PSOccurrence class. This attribute is also available and changeable in the dynamic BOMLine class.

The Tranformation Matrix is stored in the Transformation attributes of the PSOccurrence class. This attribute is also available and changeable in the dynamic BOMLine class.

CHAPTER 4

Customization Options

Preparing Customization Tasks

To customize the CMI RII software an Eclipse plugin has to be implemented. See *Teamcenter Rich Client Customization Programmer's Guide* for more details to set up an customization environment.

In the following sections the extension points are described with the following table:

Name:	**CMIRIICustomGetExample**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetExample**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**String getExample(Object exampleObject);**
Description:	**This interface function is called to retrieve example information.** **exampleObject** **The object (TCComponentDataset) to get the example from**

This example extension point can be implemented by adding the following xml tag in the plugin section of the **plugin.xml** file of your customization plugin.

```
…
<plugin>
…
<extension point="com.tsystems.cmi.r2.interfaces.CMIRIICustomGetExample">
  <customIsEmbedded class="com.xxx.custom.MyCustomGetExample"/>
</extension>
…
```

The class **com.xxx.custom.MyCustomGetExample** must implement the interface:

com.tsystems.cmi.r2.interfaces.custom.ICustomGetExample

The class **MyCustomGetExample** can then look like the following code fragment:

```
package com.xxx.custom;

import com.teamcenter.rac.kernel.TCComponentItemRevision;
import com.teamcenter.rac.kernel.TCException;
import com.tsystems.cmi.r2.interfaces.custom.ICustomGetExample;
```

```
public class MyCustomGetExample implements ICustomGetExample
{
  public String getExample(Object object)
  {
    if (object instanceof TCComponentItemRevision)
    {
      TCComponentItemRevision itemRev = (TCComponentItemRevision)object;
      if (itemRev != null)
      {
        try
        {
          return itemRev.getStringProperty("object_desc");
        }
        catch (TCException e)
        {};
      }
    }
  return "";
  }
}
```

Setting up CATIA environment

With the following environment variables the CMI Configuration File can be used for setting the CMI CATIA environment.

```
set CMI_CONFIGURABLE_NODE_BEHAVIOR=ON
set CMI_CONFIGURATION_FILE=<path>\CMICatiaV5Config.xml
```

The following example shows a CMI Configuration File with environment variables, part categories, configurable behaviors, and user defined properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CMIConfigTopics SYSTEM "CMICatiaV5Config.dtd">
<CMIConfigTopics>
  <EnvironmentVariables>
    <Variable Name="CMI_CREATE_SPECPARTS_IN_UPDATE" Value="ON"/>
  </EnvironmentVariables>
  <PartCategories Default="NOTSET" UserMustSelectNonDefault="ON">
    <PartCategory Name="PART_CATEGORY1" DisplayName="[Category1]" />
    <PartCategory Name="NOTSET" DisplayName="[**NOT SET**]" />
  </PartCategories>
  <ConfigurableBehaviors>
    <ConfigurableBehavior UniqueID = "EmbeddedNode_Run_Ignore">
      <BehaviorType>EmbeddedNodeBehavior</BehaviorType>
      <PartNumberPrefix>Run</PartNumberPrefix>
      <Behavior>IgnoreNode</Behavior>
    </ConfigurableBehavior>
  </ConfigurableBehaviors>
  <UserDefinedProperties>
    <UserDefinedProperty Name="CustomerProp1" />
  </UserDefinedProperties>
</CMIConfigTopics>
```

Used data types for customization

The following data types are used in the extension point interface:

IPdmNameValueSet	
`void set(String name, String value);` Sets a new or overwrites an existing name value pair.	
`boolean containsName(String name);` Returns true if the name is contained in the name value pair.	
`String getValue(String name);` Returns the value for the given name.	
`Vector<String> getNames();` Returns all names included in the set.	
IPdmUserDefProperties	
`void set(String name, String display, String value);` Sets a new or overwrites an existing user defined property.	
`boolean containsName(String name);` Returns true if the name is contained in the user defined properties.	
`String getValue(String name);` Returns the value for the given name.	
`String getDisplay(String name);` Returns the display for the given name.	
`Vector<String> getNames();` Returns all names included in the user defined property.	
IPdmStatus	
`void addMessage(String message);` Adds a message to the Status object.	
`void setWarningStatus();` Sets the warning status to true.	

Show Teamcenter meta data in CATIA

The *More* button in the CMI Info command allows to retrieve realtime information about the selected CATIA V5 item from Teamcenter. By default the *Get Item Info* dialogs define the information shown.

You can customize the information that is displayed by implementing the following customization points:

Customization: Show Teamcenter meta data in CATIA V5

To retrieve information from the Item Revision object, implement the following extension point:

`Name:`	**`CMIRIICustomGetPartInfo`**
`Plugin ID:`	**`com.tsystems.cmi.r2.interfaces`**
`Interface:`	**`ICustomGetPartInfo`**

Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void getPartInfo(Object object, Vector<String> infoNames, Vector<String> infoValues);**
Description:	**This interface function is called to retrieve Part/Item information for CATIA (CMI Info Dialog).** **Object the object (TCComponentItemRevision) to get the info from.** **infoNames the return vector with the names.** **infoValues the return vector with the values.**

To retrieve information from the dataset object, implement the following extension point:

Name:	**CMIRIICustomGetDatasetInfo**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetDatasetInfo**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void getDatasetInfo(Object object, Vector<String> infoNames, Vector<String> infoValues);**
Description:	**This interface function is called to retrieve Dataset information for CATIA (CMI Info Dialog).** **object the object (TCComponentDataset) to get the info from.** **infoNames the return vector with the names.** **infoValues the return vector with the values.**

To retrieve information from the Assembly Structure Relation, implement the following extension point:

Name:	**CMIRIICustomGetItemRelInfo**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetItemRelInfo**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void getItemRelInfo(Object object,**

	Vector<String> infoNames, Vector<String> infoValues);
Description:	This interface function is called to retrieve Assembly Structure Relation information for CATIA (CMI Info Dialog). Object the object (TCComponent with type PSOccurence) to get the info from. infoNames the return vector with the names. infoValues the return vector with the values.

Standard Properties in CATIA V5

In CATIA V5 standard properties (Revision, Definition, Nomenclature and Description) can be set from CMI RII. The property values could be changed by the user and all changed properties are sent back to Teamcenter during update. Standard properties may be set/stored for the TCComponentItemRevision class or TCComponentDataset class.

Customization: Sending standard attributes to CATIA V5

To customize the standard properties getter implement the following extension points:

Name:	CMIRIICustomGetCatiaDefinition
Plugin ID:	com.tsystems.cmi.r2.interfaces
Interface:	ICustomGetCatiaDefinition
Package:	com.tsystems.cmi.r2.interfaces.custom
Function:	String getCatiaDefinition(Object object);
Description:	This interface function is called to retrieve CATIA Definition from Teamcenter. object the object (TCComponentItemRevision or TCComponentDataset) to get the info from. Returns the CATIA Definition.

Name:	CMIRIICustomGetCatiaDescription
Plugin ID:	com.tsystems.cmi.r2.interfaces
Interface:	ICustomGetCatiaDescription
Package:	com.tsystems.cmi.r2.interfaces.custom
Function:	String getCatiaDescription(

	Object object);
Description:	This interface function is called to retrieve CATIA Description from Teamcenter. object the object (TCComponentItemRevision or TCComponentDataset) to get the info from. Returns the CATIA Description.

Name:	CMIRIICustomGetCatiaNomenclature
Plugin ID:	com.tsystems.cmi.r2.interfaces
Interface:	ICustomGetCatiaNomenclature
Package:	com.tsystems.cmi.r2.interfaces.custom
Function:	String getCatiaNomenclature(Object object);
Description:	This interface function is called to retrieve CATIA Nomenclature from Teamcenter. object the object (TCComponentItemRevision or TCComponentDataset) to get the info from. Returns the CATIA Nomenclature.

Name:	CMIRIICustomGetCatiaRevision
Plugin ID:	com.tsystems.cmi.r2.interfaces
Interface:	ICustomGetCatiaRevision
Package:	com.tsystems.cmi.r2.interfaces.custom
Function:	String getCatiaRevision(Object object);
Description:	This interface function is called to retrieve CATIA Revision from Teamcenter. object the object (TCComponentItemRevision or TCComponentDataset) to get the info from. Returns the CATIA Revision.

Customization: Receiving user defined attributes from CATIA V5

It is possible to receive changed standard properties from CATIA V5. These attributes may be saved back in Teamcenter.

To customize the standard properties setter implement the following extension points:

Name:	**CMIRIICustomSetCatiaDefinition**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomSetCatiaDefinition**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void setCatiaDefinition(Object object, String definition);**
Description:	**This interface function is called to set CATIA Definition to Teamcenter.** **object** **the object (TCComponentItemRevision or TCComponentDataset) to set the info for.** **definition** **the definition to set in Teamcenter.**

Name:	**CMIRIICustomSetCatiaDescription**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomSetCatiaDescription**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void setCatiaDescription(Object object, String description);**
Description:	**This interface function is called to set CATIA Description to Teamcenter.** **object** **the object (TCComponentItemRevision or TCComponentDataset) to set the info for.** **description** **the description to set in Teamcenter.**

Name:	**CMIRIICustomSetCatiaNomenclature**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomSetCatiaNomenclature**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void setCatiaNomenclature(Object object, String nomenclature);**
Description:	**This interface function is called to set CATIA**

	Nomenclature to Teamcenter. object the object (TCComponentItemRevision or TCComponentDataset) to set the info for. nomenclature the nomenclature to set in Teamcenter.

Name:	CMIRIICustomSetCatiaRevision
Plugin ID:	com.tsystems.cmi.r2.interfaces
Interface:	ICustomSetCatiaRevision
Package:	com.tsystems.cmi.r2.interfaces.custom
Function:	void setCatiaRevision(Object object, String revision);
Description:	This interface function is called to set CATIA Revision to Teamcenter. object the object (TCComponentItemRevision or TCComponentDataset) to set the info for. revision the revision to set in Teamcenter.

User Defined Properties in CATIA V5

In CATIA V5 you can add user-defined properties to the standard CATIA V5 properties form (Added Properties).

CMI provides two extension points to work with such user defined properties.

It is possible to send user defined properties from Teamcenter to CATIA V5 and display these properties within the standard properties dialog (see Figure 3). The property values could be changed by the user and all changed properties are sent back to Teamcenter during update.

It is not possible to define new properties in CATIA V5 dialog and save them back to Teamcenter.

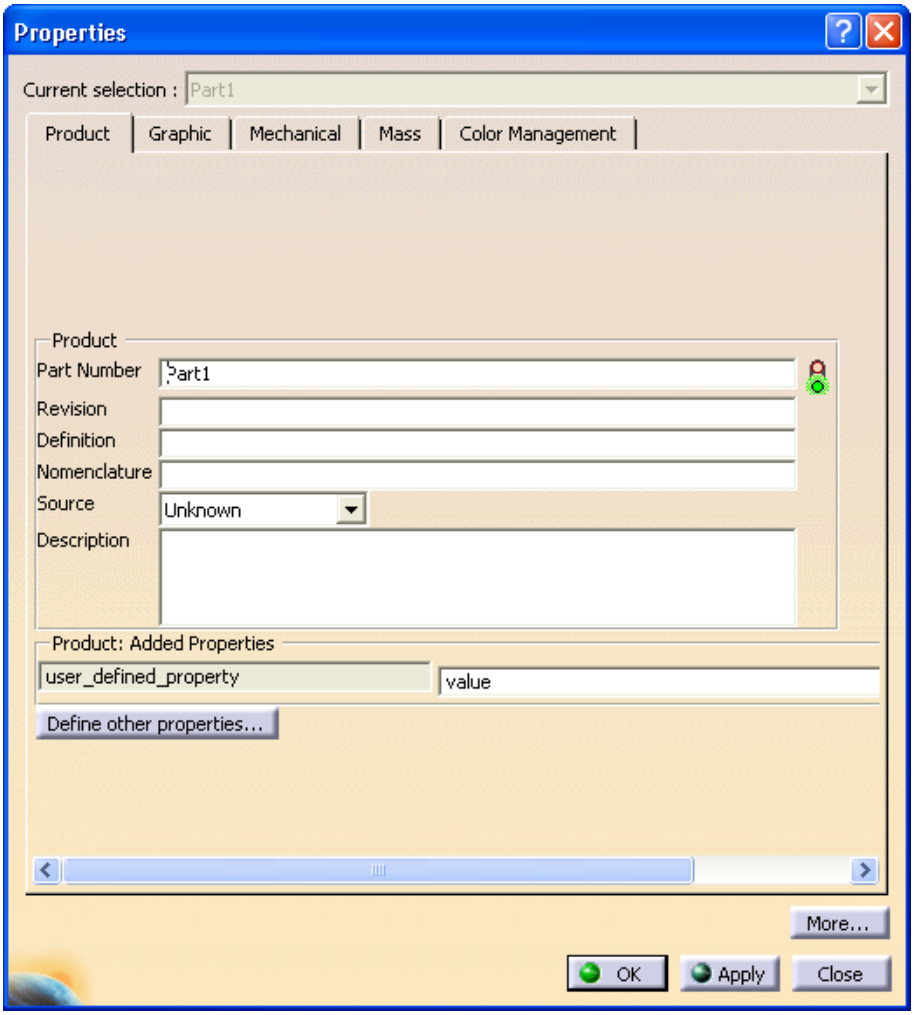

Figure 3: User defined properties

Customization: Sending user defined attributes to CATIA V5

It is possible to send user defined attributes to CATIA V5 and display them within the standard CATIA V5 properties dialog. Therefore you have to implement the following extension points:

Name:	**CMIRIICustomGetUserDefinedAttributes**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetUserDefinedAttributes**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void getUserDefinedAttributes(** **Object object,** **IPdmUserDefProperties userDefProps);**
Description:	**This interface function is called to retrieve user defined attributes from Teamcenter.** **object** **the object (TCComponentItemRevision or TCComponentDataset) to get the info from.** **userDefProps** **a named value pair object to return the user defined**

	properties.

This extension point can be used with the TCComponentItemRevision class or TCComponentDataset class.

Customization: Receiving user defined attributes from CATIA V5

It is possible to receive changed user defined properties from CATIA V5. These attributes may be saved back in Teamcenter. Therefore you have to implement the following extension point.

Name:	**CMIRIICustomSetUserDefinedAttributes**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomSetUserDefinedAttributes**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean setUserDefinedAttributes(** **Object object,** **IPdmUserDefProperties userDefProps);**
Description:	**This interface function is called to set user defined attributes from CATIA in Teamcenter. Returns true if the object was updated or changed.** **object** **the object (TCComponentItemRevision or TCComponentDataset) to set the info for.** **userDefProps** **a named value pair object with the user defined properties.**

If newly created user defined properties within CATIA V5 standard dialog should be stored in Teamcenter, they have to be declared in the CMI_CONFIGURATION_FILE.

The declaration in the CMI_CONFIGURATION_FILE of newly created properties in CATIA which should be stored in Teamcenter looks as follows:

```
<UserDefinedProperties>
      <UserDefinedProperty Name="CustomerProp1" />
      <UserDefinedProperty Name="CustomerProp2" />
</UserDefinedProperties>
```

Configurable Behaviors in CATIA V5

It is possible to configure the behavior of CATIA while update & synchronize. Dependend on a given prefix of the partnumber in a CATProduct/Component it is possible to force the update to:

- Ignore a Component.
- Ignore the CATProduct/Component and it's subtree.
- Create a special relation in Teamcenter.

This has to be configured in the CMI_CONFIGURATION_FILE. The scheme looks as follows:

```
<CMIConfigTopics>
…
      <ConfigurableBehaviors>
            <ConfigurableBehavior UniqueID = "[Unique ID]">
                  <BehaviorType>[Behavior Type]</BehaviorType>
                  <PartNumberPrefix>[Prefix]</PartNumberPrefix>
                  <Behavior>[Behavior]</Behavior>
            </ConfigurableBehavior>
      </ConfigurableBehaviors>
…
</CMIConfigTopics>
```

There can be multiple tags **<ConfigurableBehavior>** in the tag **<ConfigurableBehaviors>**

The tag **<UniqueID>** has to contain a value which has to be unique in this file.

The following options exist to define which CATIA components a configurable behavior shall apply to:

Based on its part number -

<PartNumberPrefix>Spec_</PartNumberPrefix>

The behavior applies to components whose part number begins with “Spec_”

An empty PartNumberPrefix makes the behavior apply to any component.

Based on its product type -

<ProductType>ElecWireGroup</ProductType>

The behavior applies to components of the type ElecWireGroup. To help with configuration, the Product type is shown in the CMI Info dialog.

Based on its instance name -

<InstanceNamePrefix>XY_</InstanceNamePrefix>

The behavior applies to components where the instance name begins with “XY_”.

When these tags are combined, a component must match the requirement of either tag.

The following combinations of values are valid:

No.	BehaviorType	Behavior
1	EmbeddedNodeBehavior	SkipNode
2	EmbeddedNodeBehavior	ReferenceGeometry
3	EmbeddedNodeBehavior	DeepSkipNode
4	EmbeddedNodeBehavior	IgnoreNode
5	ProductNodeBehavior	IgnoreNode

Descriptions of the behaviors:

1. **SkipNode:**

The component is skipped and its children are instantiated as a direct child of the CATProduct/Assembly that contains the component.

2. **DeepSkipNode:**

The component and all subsequent components beneath it are skipped, up to the next regular CATProduct/CATPart.

3. **IgnoreNode:**

The Component and its substructure is ignored in Teamcenter.

4. **ReferenceGeometry**:

The Component is skipped and its children are instantiated in Teamcenter with a special Reference relation instead of the standard Assembly relation. So the substructure of this component will not be part of the BOM.

5**. IgnoreNode (Product):**

CATProduct is ignored in Teamcenter. This may result in broken links as the Product is not provided by Teamcenter during a load.

Configurable Check In/Out Dialog in CATIA V5

It is possible to configure the additional columns of the Check In/Out dialog in CATIA.

This has to be configured in the CMI_CONFIGURATION_FILE. The scheme looks as follows:

```
<CheckInOutAttributes>
      <CheckInOutAttribute Name="[Name]"
                           DisplayName="[Display Name]"
                           ColumnWidth="[Column Width]" />
</CheckInOutAttributes>
```

The [Name] is an alias name, which has to be recognized in Teamcenter. The following patterns of the alias are supported by the default implementation:

D#attributeName, e.g. “D#object_string”

I#attributeName, e.g. “I#object_string”

where “D” stands for “Dataset” and “I” stands for “Item” followed by the hash sign and the attribute name. In these cases the attribute values from dataset or item revision are read and send as result.

E.g.

```
<CheckInOutAttributes>
      <CheckInOutAttribute Name="LockedBy"
                           DisplayName="Checked-Out By"
                           ColumnWidth="15" />
      <CheckInOutAttribute Name="D#object_string"
                           DisplayName="Object String Dataset"
                           ColumnWidth="15" />
      <CheckInOutAttribute Name="I#object_string"
                           DisplayName="Object String Revision"
                           ColumnWidth="15" />
</CheckInOutAttributes>
```

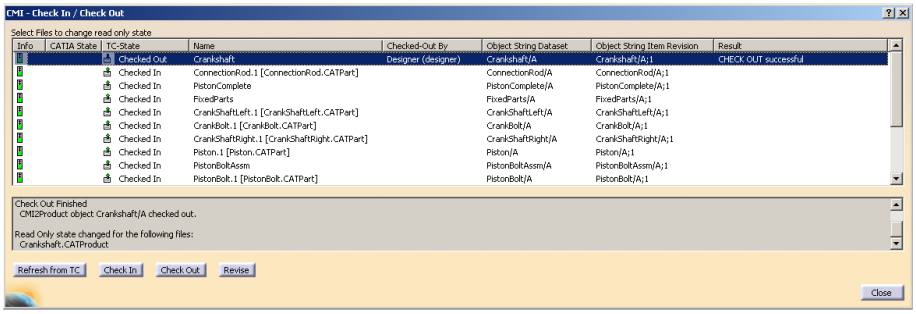

Figure 4: Check In/Check Out dialog

Customization: Receiving additional column attributes from CATIA V5

You have to implement the following extension point in order to get additional attributes.

Name:	**CMIRIICustomGetCheckInOutDlgAdditionalInfo**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetCheckInOutDlgAdditionalInfo**
Package:	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean getCheckInOutDlgAdditionalInfo(Object dataset, Object itemRevision, IPdmNameValueSet additionalAttributes)**
Description:	**This interface function is called to retrieve additional attributes for CATIA (Check In/Out Dialog). For the supported patterns of the default implementation please see above.** **dataset** **dataset of the CATIA object** **itemRevision** **related item revision of the CATIA object** **additionalAttributes** **the return vector with the values**

Working with CATIA V5 Released Cache

CMI supports the use of CGR files in the released cache of CATIA V5. For this purpose the Teamcenter customization has to store the CGR files of CATIA models in Teamcenter. During “To Catia” these CGR files are copied to the Released Cache instead of the CATIA models to the exchange map. In CATIA V5 the CGR files are loaded in visualization mode. For each file in the workbench CMI decides whether to copy the standard file or the CGR file.

Teamcenter options:

Edit → Options …

In the CMI RII option panel set “Transfer CGR Files to CATIA” to “Only CGR” or “CGR+geometry”.

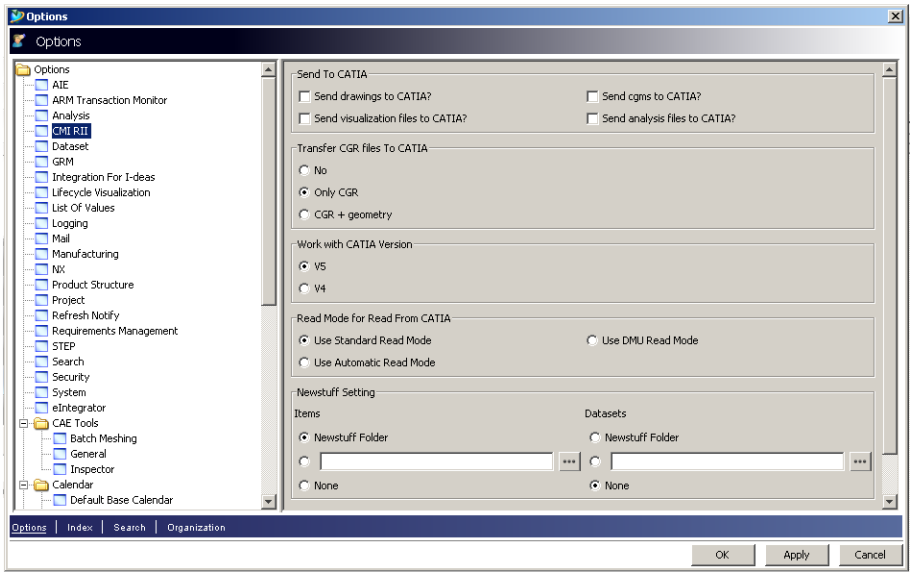

Figure 5: Teamcenter CMI RII options

No	Only CATParts. cgr files are not used.
Only CGR	If a public CATPart has a corresponding cgr file CMI transfers only the CGR file to CATIA.
CGR + geometry	If a public CATPart has a corresponding cgr file CMI transfers both to CATIA: the cgr file and the CATPart.

CATIA Options:

Tools→Options …

Set **"Work with the Cache System"** to **"On"** and set the correct path to the released cache.

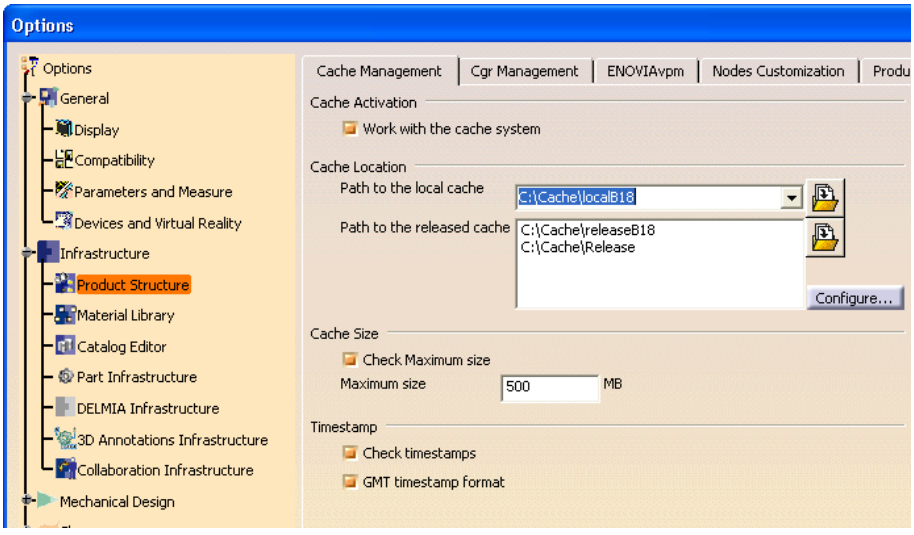

Figure 6: CATIA Cache Management options

These are the necessary preconditions that a CGR file is copied to the released cache:

- Work with the cache system in CATIA V5 is enabled.
- The Released cache is set and exists.
- The CATIA V5 configuration variable **CMI_USERELEASEDCACHE** is set to "ON".

- The CMI RII Teamcenter preference “Transfer CGR-File to CATIA V5” is set to “Only CGR” or “CGR + geometry”.
- The CATIA dataset must be loaded read only in CATIA.
  No Read access in Teamcenter or CMIUseChangeableMeansOwner (see Chapter 5) is set to 1, and the dataset is not checked out.
- A cgr dataset exists for the CATIA dataset.

Optional settings in the CATIA Environment:

To use a special released cache directory out of the list in CATIA the **CMI_RELEASEDCACHEDIR** setting is optional. If not set the first released cache in list is used.

```
set CMI_RELEASEDCACHEDIR=<Path to released cache>
```

If you have set the CATIA environment **CMI_RELEASEDCACHEDIR** one setting in the list of “Path to the released cache” has to be exactly identical. **CMI_RELEASEDCACHEDIR** should only be used if you have set more than one “Path to the released cache”. If not, CMI tries to add the **CMI_RELEASEDCACHEDIR** to the list of released Cache directories. Therefore the **CMI_RELEASEDCACHEDIR** and all directories in the list have to exist on the file system.

It is possible to use environment variables in **CMI_RELEASEDCACHEDIR**, like **C:\ReleasedCache\${username}**.

On UNIX you have to mask the $ sign**: /ReleasedCache/\${USER}**

To use temporary CATIA components which contain the related CGR as shape representation, instead of the original CATPart:

```
set CMI_CREATETEMPCGRCOMP=ON
```

If a CATProduct doesn’t use the CATPart before, CMI have to build in the CATPart in the CATIA Product structure. Therefore the original CATPart is needed. In this case CMI also transfers the CATPart to the CMIXMAP even if a CGR file exists.

If the environment **CMI_CREATETEMPCGRCOMP** is set and the parent CATProducts are loaded in “read only” mode the CGR file is loaded into the structure as TMP_ component.

Design Table Support

Design Tables are managed in Teamcenter by the Synchronize command in CATIA V5.

Configuration

If the Teamcenter preference variable **"CMIWorkWithDesignTables"** is set to **"1"**, during *To Catia* all CATIA CATPart and CATProduct datasets are expanded for Design Tables and the relevant Design Tables are transferred to CATIA V5. There is a performance impact.

Optional settings in the CATIA Environment:

To use Design Tables for CATProducts:

```
set CMI_USE_DTFORPRODUCT=ON
```

To prevent Design Tables for CMI-Archives.

```
set CMI_DISABLE_DT_IN_ARCHIVE=ON
```

The following extension points are used to get related Design Tables for a CATPart/CATProduct or to relate a Design Table to a CATPart/CATProduct.

Customizable extension points

Add a Design Table to a CATIA model.

Name:	**CMIRIICustomAddDesignTable**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomAddDesignTable**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void addDesignTable(Object itemRevision, Object modelDataset, Object designTableDataset, String designTableFileName, IPdmNameValueSet catiaProperties, IPdmStatus statusObject);**
Description:	**This interface function is called to add a DesignTable to a Model object.** **itemRevision** **the item revision object (TCComponentItemRevision).** **modelDataset** **the model object (TCComponentDataset).** **designTableDataset** **the design table object (TCComponentDataset).** **designTableFileName** **the file name of the DesignTable to add.** **catiaProperties** **additional Information from CATIA.** **statusObject** **the status object to store messages for CATIA.**

Remove a Design Table from a CATIA model.

Name:	**CMIRIICustomRemoveDesignTable**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomRemoveDesignTable**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void removeDesignTable(Object itemRevision, Object modelDataset, Object designTableDataset, String designTableFileName, IPdmStatus statusObject);**
Description:	**This interface function is called to remove a DesignTable from a Model object.** **itemRevision**

	the item revision object (TCComponentItemRevision). **modelDataset** **the model object (TCComponentDataset).** **designTableDataset** **the design table object (TCComponentDataset).** **designTableFileName** **the file name of the DesignTable to add.** **statusObject** **the status object to store messages for CATIA.**

Retrieve all Design Table file names for a CATIA model.

Name:	**CMIRIICustomGetKnownDesignTableFileNames**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetKnownDesignTableFileNames**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void getKnownDesignTableFileNames(** **Object itemRevision,** **Object modelDataset,** **Object designTableDataset,** **String designTableFileName,** **IPdmNameValueSet catiaProperties,** **IPdmStatus statusObject);**
Description:	**This interface function is called to retrieve the file names of all design tables for the model dataset.** **itemRevision** **the item revision object (TCComponentItemRevision).** **modelDataset** **the model object (TCComponentDataset).** **designTableDataset** **the design table object (TCComponentDataset).** **designTableFileName** **the file name of the DesignTable to add.** **catiaProperties** **additional information from CATIA.** **statusObject** **the status object to store messages for CATIA.**

Retrieve all Design Table Datasets for a CATIA model.

Name:	**CMIRIICustomGetDesignTablesForNames**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetDesignTablesForNames**

Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**Objcet[] getDesignTablesForNames(Object itemRevision, Object modelDataset, String[] designTableNames, IPdmStatus statusObject);**
Description:	**This interface function is called to retrieve the design tables for the given names. Returns the design tables as array with the format TCComponentDataset[].** **iemRevision** **the item revision object (TCComponentItemRevision).** **modelDataset** **the model object (TCComponentDataset).** **designTableNames** **the list with the requested Design Table file names.** **statusObject** **the status object to store messages for CATIA.**

MML Support

MMLs can be managed in Teamcenter.

Installation

This use case requires additional CMI CATIA library:

Windows:	CMIExt.dll
Solaris:	libCMIExt.so
HP-UX:	libCMIExt.sl
AIX:	libCMIExt.a
IRIX:	libCMIExt.so

This library is not part of the CMI RII product. Please contact *cmi_support@t-systems.com* to get more information.

Copy this library to the binary path of the CMI CATIA installation, e.g. <CMIRII_Installation_Dir>\CMIRIICATV5_R19_V230\intel_a\code\bin\

Configuration

Teamcenter preference:

The CMIDependentTypeList preference defines the MML types which will be processed, default is **"CCP"** and **"Design".**

Set this preference if you want to use other link types than **CCP** or **Design**.

CATIA V5 Environment settings:

set CMI_GETPOINTEDDOCUMENTS=ON, default is OFF

Set this environment to "ON" to provide the information about referenced CATIA files to Teamcenter.

set CMI_ENABLE_CMIEXTERNALDOCCMD=ON, default is OFF

Set this environment to “ON” to enable the “Get Referenced Geometries” button in CATIA V5.

set CMI_ENABLE_CMIGETDEPBYCMD=ON, default is OFF

Set this environment to “ON” to enable the “Get Depended By Geometries” button in CATIA V5 (Teamcenter Customizing is required due to version ambiguities.)

Customizable extension points

Manage dependencies during Update/Synchronize/Create/SaveAs:

Name:	**CMIRIICustomProcessDependency**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomProcessDependency**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void processDependency(Object sourceRevision, Object sourceDataset, Vector<String> targetRevisionList, Vector<String> targetDatasetList, Vector<String> targetDocumentList, Vector<String> targetTypesList, IPdmStatus statusObject);**
Description:	**This interface function is called to handle the dependency information from CATIA in Teamcenter.** **sourceRevision** **the source revision object (TCComponentItemRevision).** **sourceDataset** **the source dataset object (TCComponentDataset).** **targetRevisionList** **the target revision list with the target part UIDs.** **targetDatasetList** **the target dataset list with the target dataset UIDs.** **targetDocumentList** **the target document file name.** **targetTypesList** **the link types.** **statusObject** **the status object to add messages for CATIA.**

Get Referenced Geometries:

Name:	**CMIRIICustomGetDependentObjects**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**

Interface:	**ICustomGetDependentObjects**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**Object[] getDependentObjects(Object dataset, IPdmStatus statusObject);**
Description:	**This interface function is called to retrieve the dependend Model objects. Returns the related dataset objects in an Object array of type TCComponentDataset[].** **dataset** **the dataset (TCComponentDataset) to search the dependent objects for.** **statusObject** **the status object to add messages for CATIA.**

Get Depended By Geometries:

Name:	**CMIRIICustomGetDependentByObjects**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetDependentByObjects**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**Object[] getDependentByObjects(Object dataset, IPdmStatus statusObject);**
Description:	**This interface function is called to retrieve the dependent by Model objects. Returns the related dataset objects in an Object array of type TCComponentDataset[].** **dataset** **the dataset (TCComponentDataset) to search the dependent by objects for.** **statusObject** **the status object to add messages for CATIA.**

When you are using “Get Depended By Geometries“, you have to make sure your implementation of **CMIRIICustomGetDependentByObjects** filters the depended on by files. It is not possible to load multiple versions of one file into CATIA V5.

Transfer Model Infos to Teamcenter

Weight properties (inertia)

Inertia properties – eg. Mass – can be read from CATIA V5 and stored in Teamcenter.

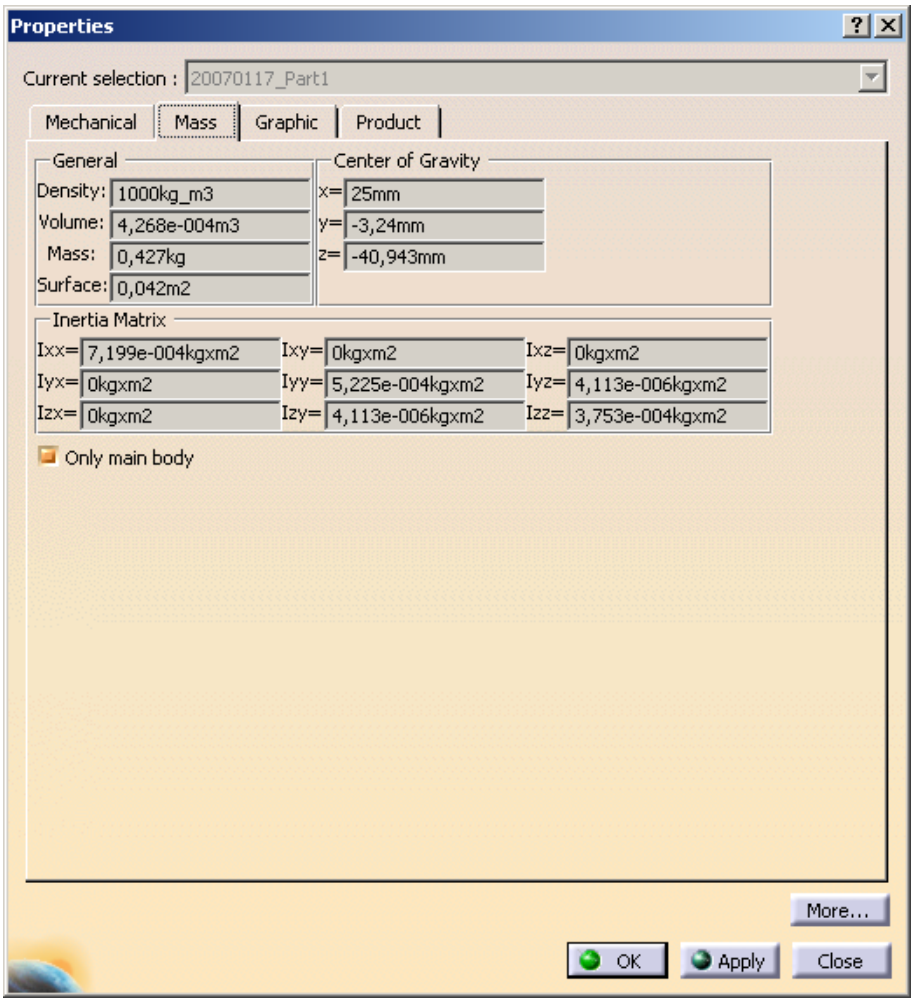

Figure 7: CATIA *Mass* properties

Configuration

CATIA V5 Environment settings:

In order to read these properties from CATIA, the following variable has to be set in the CATIA V5 environment:

`set CMI_READ_INERTIA=ON` (Inertias from CATProducts and CATParts are sent to Teamcenter.)

`set CMI_READ_INERTIA=ONLY_CATPART` (Only the inertias from the CATParts will sent to Teamcenter. This may improve the performance especially in case of large assemblies.)

The inertias can be confined to the main bodies of the CATParts by setting:

set CMI_CONFINE_INERTIA_TO_MAINBODIES=ON

CATIA V5 Version information

CATIA V5 Version information is read in CATIA V5 and stored in Teamcenter.

The storage of the inertia and CATIA version information can be customized in Teamcenter by implementing the **CMIRIICustomSetModelInfos** extension point for the CATProduct and/or the CATPart.

Customizable extension points

Store Model Infos in Teamcenter

Name:	**CMIRIICustomSetModelInfos**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomSetModelInfos**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void setModelInfos(Object revision, Object dataset, IPdmNameValueSet modelInfos);**
Description:	**This interface function is called to set the model infos to Teamcenter.** **revision** **the related revision object (TCComponentItemRevision).** **dataset** **the releated dataset object (TCComponentDataset).** **modelInfos** **the model infos to set.**

The extension point is called in the course of Update, Create and similar actions.

CMI stores the model infos information in the CMI2CatiaInfoForm (CATIA Version Information) and CMI2CadAttrForm (inertia information).

The following named values are provided to the API in the modelInfos parameter:

CATIA Version Information

`CATIA_RELEASE, CATIA_SERVICE_PACK, CATIA_HOTFIX`

Mass

`INERTIA_MASS, INERTIA_VOLUME, INERTIA_DENSITY, INERTIA_AREA`

Position of the center of gravity

`INERTIA_POSITION_0, INERTIA_POSITION_1, INERTIA_POSITION_2`

Inertia matrix

```
INERTIA_MATRIX_0, INERTIA_MATRIX_1, INERTIA_MATRIX_2,
INERTIA_MATRIX_3, INERTIA_MATRIX_4, INERTIA_MATRIX_5,
INERTIA_MATRIX_6, INERTIA_MATRIX_7, INERTIA_MATRIX_8
```

Components of principal axes

```
INERTIA_COMPONENTS_0, INERTIA_COMPONENTS_1, INERTIA_COMPONENTS_2,
INERTIA_COMPONENTS_3, INERTIA_COMPONENTS_4, INERTIA_COMPONENTS_5,
INERTIA_COMPONENTS_6, INERTIA_COMPONENTS_7, INERTIA_COMPONENTS_8
```

Principal moments values

`INERTIA_VALUES_0, INERTIA_VALUES_1, INERTIA_VALUES_2`

All the coordinates are expressed with respect to the product axis system.

If the product is made of non homogeneous material the output density is set to -1.

Used units:

- Mass kg (kilogram)
- Inertia area m^2 (square meter)
- Inertia volume m^3 (cubic meter)
- Center of gravity position m (meter)

- Inertia matrix kg m^2 (square kilogram meter)
- Principal moments kg m^2 (square kilogram meter)
- Density kg/m^3 (kilogram per cubic meter)

Set Bom Type of new CATIA files by Teamcenter customization

In the Synchronize dialog all new CATIA files are presented with a Bom Type(Bom/Non-Bom/Not Set). This Bom Type is initialized by a default and can be changed by the user. Depending on the CATIA V5 configuration variable **CMI_GET_BOMTYPE_FROM_TC=ON** the values for the Bom Type are fetched from Teamcenter.

With Standard CMI the Bom Type is set to "", i.e. the Bom Type in CATIA remains unchanged.

If you want to customize the setting of the Bom Type you have to implement the extension point:

Name:	**CMIRIICustomGetBomType**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetBomType**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**String getBomType(IPdmNameValueSet nvSet, IPdmNameValueSet userDefNvSet, IPdmNameValueSet parentNvSet, IPdmNameValueSet parentUserDefNvSet);**
Description:	**This interface function is called to retrieve the bom type for a new CATPart object in CATIA. Returns the bom type to use in CATIA.** **Allowed values for the return-value:** **NOT_SET - The User must set the correct value in CATIA.** **BOM - The model will become a BOM type.** **NOT_BOM - The model will become a NON-BOM type.** **For all other values the Bom Type in CATIA remains unchanged.** **nvSet** **information from the CATPart to be created in CATIA.** **userDefNvSet** **user defined attributes for the CATPart to be created in CATIA.** **parentNvSet** **information from the parent of the CATPart to be created in CATIA.** **parentUserDefNvSet** **user defined attributes for the parent object.**

The nvsSet contains the following CATIA attributes for the new CATIA file, if available:

PARTNUMBER, FILENAME, and NOMENCLATURE

The userDefNvSet contains the user defined attributes for the new CATIA file, if existent.

The parentNvSet contains the following attributes for the parent object, i. e. the Assembly, if existent:

PARTNUMBER, NOMENCLATURE, DB_NAME, OBID, and CLASSNAME

The parentUserDefNvSet contains the user defined attributes for the parent object, if existent.

Validation of CMI Archive names

If **CMI_ENABLE_ARCHIVE_VALIDATION=ON** is set in the CATIA environment the CMIArchive Validation in Teamcenter is called before CMIArchive Create/Update. The default behavior is, that all validations are successful. The following extension point can be used for customization of the validation.

Name:	**CMIRIICustomValidateArchiveName**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidateArchiveName**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validateArchiveName(String parentPartNumber, IPdmNameValueSet objInfos, IPdmStatus statusObject);**
Description:	**This interface function is called to validate the archive names. Return true it the validation is ok, false if the validation fails.** **parentPartNumber** **the part number of the CATProduct which assembles the Archive.** **objInfos** **a name value set with the information to validate.** **statusObject** **the status object to store messages for CATIA.**

Validation before running the CATIA Synchronize command

If **CMI_ENABLE_VALIDATE_BEFORE_UPD=ON** is set in the CATIA environment the actions of synchronize can be validated by Teamcenter customization. The default behavior is, that all validations are successful. The following extension points can be used for customization of the validation.

Validation of object creation:

Name:	**CMIRIICustomValidateCreateV5Object**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidateCreateV5Object**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validateCreateV5Object(IPdmNameValueSet parentObjInfos, IPdmNameValueSet objInfos, IPdmStatus statusObject);**
Description:	**This interface function is called to validate the creation of a V5 object in Teamcenter. Return true it the validation is ok, false if the validation fails.** **parentObjInfos** **a name value set with the information of the parent object.** **objInfos** **a name value set with the information of the object to validate.** **statusObject** **the status object to store messages for CATIA.**

Validation of object update:

Name:	**CMIRIICustomValidateUpdateV5Object**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidateUpdateV5Object**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validateUpdateV5Object(IPdmNameValueSet parentObjInfos, IPdmNameValueSet objInfos, IPdmStatus statusObject);**
Description:	**This interface function is called to validate the update of a V5 object in Teamcenter. Return true it the validation is ok, false if the validation fails.** **parentObjInfos** **a name value set with the information of the parent object.** **objInfos** **a name value set with the information of the object to validate.** **statusObject** **the status object to store messages for CATIA.**

Validation of object prepare:

Name:	**CMIRIICustomValidatePrepareV5Object**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidatePrepareV5Object**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validatePrepareV5Object(IPdmNameValueSet parentObjInfos, IPdmNameValueSet objInfos, IPdmStatus statusObject);**
Description:	**This interface function is called to validate the prepare of a V5 object in Teamcenter. Return true it the validation is ok, false if the validation fails.** **parentObjInfos** **a name value set with the information of the parent object.** **objInfos** **a name value set with the information of the object to validate.** **statusObject** **the status object to store messages for CATIA.**

Validation of object use:

Name:	**CMIRIICustomValidateUseV5Object**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidateUseV5Object**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validateUseV5Object(IPdmNameValueSet parentObjInfos, IPdmNameValueSet objInfos, IPdmStatus statusObject);**
Description:	**This interface function is called to validate the use of a V5 object in Teamcenter. Return true it the validation is ok, false if the validation fails.** **parentObjInfos** **a name value set with the information of the parent object.** **objInfos** **a name value set with the information of the object to validate.** **statusObject** **the status object to store messages for CATIA.**

Validation of instance attach:

Name:	**CMIRIICustomValidateAttachV5ObjectInstance**

Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidateAttachV5ObjectInstance**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validateAttachV5ObjectInstance(IPdmNameValueSet parentObjInfos, IPdmNameValueSet objInfos, IPdmStatus statusObject);**
Description:	**This interface function is called to validate the attach of a V5 instance in Teamcenter. Return true it the validation is ok, false if the validation fails.** **parentObjInfos** **a name value set with the information of the parent object.** **objInfos** **a name value set with the information of the object to validate.** **statusObject** **the status object to store messages for CATIA.**

Validation of instance update:

Name:	**CMIRIICustomValidateUpdateV5ObjectInstance**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidateUpdateV5ObjectInstance**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validateUpdateV5ObjectInstance(IPdmNameValueSet parentObjInfos, IPdmNameValueSet objInfos, IPdmStatus statusObject);**
Description:	**This interface function is called to validate the update of a V5 instance in Teamcenter. Return true it the validation is ok, false if the validation fails.** **parentObjInfos** **a name value set with the information of the parent object.** **objInfos** **a name value set with the information of the object to validate.** **statusObject** **the status object to store messages for CATIA.**

Validation of instance drop:

Name:	**CMIRIICustomValidateDropV5ObjectInstance**

Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidateDropV5ObjectInstance**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validateDropV5ObjectInstance(IPdmNameValueSet parentObjInfos, IPdmNameValueSet objInfos, IPdmStatus statusObject);**
Description:	**This interface function is called to validate the drop of a V5 instance in Teamcenter. Return true it the validation is ok, false if the validation fails.** **parentObjInfos** **a name value set with the information of the parent object.** **objInfos** **a name value set with the information of the object to validate.** **statusObject** **the status object to store messages for CATIA.**

Customization option for Create

The following customization extension points are called while creating Teamcenter Objects from CATIA.

Name:	**CMIRIICustomGetPartNumberForCreate**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetPartNumberForCreate**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**String getPartNumberForCreate(String partNumber, IPdmNameValueSet catiaInfos, IPdmNameValueSet userDefProps);**
Description:	**This interface function is called to retrieve the part number for create. Return the new part number to use for create.** **partNumber** **the actual part number.** **catiaInfos** **additional CATIA infos for the item create.** **userDefProps** **additional user defined attributes.**

Name:	**CMIRIICustomGetDatasetNameForCreate**

Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetDatasetNameForCreate**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**String getDatasetNameForCreate(Object revision, String dsType, String datasetName, String fileName);**
Description:	**This interface function is called to retrieve the dataset name for create. Return the new dataset name to use for create.** **revision** **the revision (TCComponentItemRevision) where the dataset will be attached to.** **dsType** **the dataset type.** **datasetName** **the actual dataset name.** **fileName** **the CATIA file name of the file which will be attached to the dataset.**

Name:	**CMIRIICustomGetFileNameForCreate**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetFileNameForCreate**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**String getFileNameForCreate(Object revision, Object dataset, String fileNameForCreate);**
Description:	**This interface function is called to retrieve the file name for create. Return the new file name to use for create.** **revision** **the revision (TCComponentItemRevision) where the dataset is attached to.** **dataset** **the dataset (TCComponentDataset) where the file will be referenced to.** **fileNameForCreate** **the actual file name.**

Customization option for Read

The following customization extension points are called while reading Teamcenter Objects to CATIA.

Name:	**CMIRIICustomIsComponent**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomIsComponent**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean isComponent(Object object);**
Description:	**This interface function is called to retrieve the isComponent flag for CATIA (→Item is handled as BOM Part in CATIA). The default is that if a CMI3DGeoDataset is attached to the item revision the item is handled as a BOM Part. Return true if the object is a BOM-Part.** **object the revision (TCComponentItemRevision) to get the isComponent flag for.**

Name:	**CMIRIICustomIsEmbedded**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomIsEmbedded**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean isEmbedded(Object object);**
Description:	**This interface function is called to retrieve the isEmbedded flag for CATIA (→Item is handled as CATIA Component ,no CATProduct File, in CATIA). Return true if the object should be loaded as CATIA Component.** **object the revision (TCComponentItemRevision) to get the isEmbedded flag for.**

Name:	**CMIRIICustomIsProductAttachable**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomIsProductAttachable**
Package	**com.tsystems.cmi.r2.interfaces.custom**

Function:	**boolean isProductAttachable(Object object);**
Description:	**This interface function is called to retrieve the isProductAttachable flag for CATIA. Return true if it is allowed to attach a CATProduct/BOM-CATPart Dataset to the item.** **object the revision (TCComponentItemRevision) to get the isProductAttachable flag for.**

CATProcess customization

The CATProcess handling can be adapted by the following customization points.

Customizable extension points

The customization point isProcessItem is used to detect a process element. The custom point is only called for top level items. The default implementation expands to an CMI2Process dataset which is the storage class for a CATProcess file. The default implementation returns true if a Process is found.

Name:	**CMIRIICustomIsProcessItem**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomIsProcessItem**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean isProcessItem(Object itemRevision);**
Description:	**This interface function is called to retrieve the isProcessItem flag for CATIA.** **itemRevision the item revision object (TCComponentItemRevision).**

The customization point isProductViewItem is called if a process element (see isProcessItem) expands its children. The CATIA V5 structure under the product view is added in the product view of the CATProcess in CATIA V5. The default implementation returns true.

Name:	**CMIRIICustomIsProductViewItem**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomIsProductViewItem**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean isProductViewItem(Object itemRevision);**

Description:	**This interface function is called to retrieve the isProductViewItem flag for CATIA.** **itemRevision** **the item revision object (TCComponentItemRevision).**

The customization point isResourceViewItem is called if a process element (see isProcessItem) expands its children. The CATIA V5 structure under the resource view is added in the resource view of the CATProcess in CATIA V5. The default implementation returns false.

Name:	**CMIRIICustomIsResourceViewItem**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomIsResourceViewItem**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean isResourceViewItem(Object itemRevision);**
Description:	**This interface function is called to retrieve the isResourceViewItem flag for CATIA.** **itemRevision** **the item revision object (TCComponentItemRevision).**

CATIA version check customization

The validateCatiaVersion customization point is called while sending files to CATIA. and will validate the stored CATIA version with the catia release. These customization is only called when the Teamcenter preference “CMIValidateCatiaVersion is set to 1.

The default implementation compares the given CATIA relase with the stored CATIA relase of the dataset.

Name:	**CMIRIICustomValidateCatiaVersion**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomValidateCatiaVersion**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**boolean validateCatiaVersion(Object modelObject, String catiaRelease, String catiaServicePack, String catiaHotfix);**
Description:	**This interface function is called to validate the CATIA version of the given object. Returns false if the stored CATIA release is greater than the given CATIA release;otherwise true.** **object**

	the modelObject (TCComponentDataset) to validate the CATIA version **catiaRelease** **the CATIA release** **catiaServicePack** **the CATIA service pack** **catiaHotfix** **the CATIA hotfix**

The getCatiaRelease custom point is called to retrieve the stored CATIA release for the given dataset.

Name:	**CMIRIICustomGetCatiaRelease**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetCatiaRelease**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**String getCatiaRelease(** **Object modelObject);**
Description:	**This interface function is called to get the CATIA version of the file. Returns the CATIA release of the given object.** **object** **the modelObject (TCComponentDataset) to get the CATIA version**

Mapping file customization

The following customization extension points are called while getting the mapping file in Teamcenter.

The default implementation gets the mapping dataset for the given original part number. When there are more than one dataset found a selection dialog is shown to the user.

Name:	**CMIRIICustomGetMappingFile**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetMappingFile**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**Object getMappingFile (** **String originalPartNumber);**
Description:	**This interface function is called to retrieve the mapping dataset.** **originalPartNumber** **the part number which is a search criteria** **Returns the mapping file dataset**

	(TCComponentDataset).

The default implementation gets the mapping dataset for the given item revision. When there are more than one dataset found a selection dialog is shown to the user.

Name:	**CMIRIICustomFetchMappingFile**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomFetchMappingFile**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**Object fetchMappingFile (Object itemRevision);**
Description:	**This interface function is called to retrieve the mapping dataset.** **itemRevision the item revision (TCComponentItemRevision) to get the dataset for** **Returns the mapping file dataset (TCComponentDataset).**

Customizable naming schemes for Export

The naming schema functionality for the export has to be switched on by setting the CATIA environment variable CMI_EXPORT_CUSTOMIZE_NAMING to “ON”.

The following customization extension points are called while export with a naming schema.

The default implementation can get the naming schema in the user interaction with a dialog. In order to switch on the dialog you have to set the Teamcenter preference CMIUseNamingSchemaDialog to “1”. Default value is “0”. One of the following naming schemas can be selected:

Use external names: no changes

FileName -> FileName_Revision: the revision of the object is added to the given file name

PartNumer -> PartNumber_Revision: the revision of the object is added to the given part number

FileName -> FileName_Revision and PartNumber -> PartNumber_Revision: both values are changed; see above

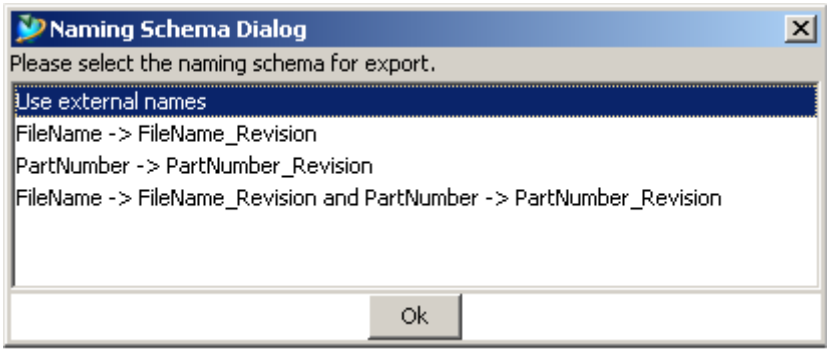

Figure 8: Naming Schema dialog

Name:	**CMIRIICustomGetExportNamingSchema**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetExportNamingSchema**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**String getExportNamingSchema (String rootPartNumber, IPdmId rootPdmId, IPdmId mappingPdmId, IPdmStatus statusObject);**
Description:	**This interface function is called to retrieve the naming schema.** **rootPartNumber** **the part number of the root part in CATUA** **rootPdmId** **the root part in CATIA** **mappingPdmId** **the mapping file if used by Teamcenter (optional)** **Returns the naming schema (String).**

The default implementation gets the reference properties for the given naming schema. There are four naming schema supported:

FileName_Revision: the revision of the object is added to the given file name

PartNumber_Revision: the revision of the object is added to the given part number

FileName_Revision_and_PartNumber_Revision: both values are changed; see above

NONE: no changes

Name:	**CMIRIICustomGetExportReferenceProperties**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetExportReferenceProperties**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**IPdmNameValueSet getExportReferenceProperties (String namingSchema, IPdmId pdmId, IPdmNameValueSet properties, IPdmStatus statusObject);**
Description:	**This interface function is called to retrieve the reference properties for the given naming schema..** **namingSchema** **the identifier of the used naming schema** **pdmId** **the BOM part; the dataset for non-BOM part**

	properties **the CATIA and the external properties. The following names are supported:** ***CatPartNumber*** **part number in CATIA (empty for Design Tables)** ***CatFileName*** **file name in CATIA** ***CatNomenclature*** **nomenclature in CATIA (optional)** ***CatRevision*** **revision in CATIA (optional)** ***CatDefinition*** **definition in CATIA (optional)** ***CatDescription*** **description in CATIA (optional)** ***ExtPartNumber*** **external part number from the mapping file (optional)** ***ExtFileName*** **external file name from the mapping file (optional)** ***ExtNomenclature*** **external nomenclature from the mapping file (optional)** ***ExtRevision*** **external revision from the mapping file (optional)** ***ExtDefinition*** **external definition from the mapping file (optional)** ***ExtDescription*** **external description from the mapping file (optional)** **statusObject** **the messages displayed in CATIA** **Returns the values for the reference (IPdmNameValueSet). The following names are supported:** ***PartNumber*** **CATIA part number** ***FileName*** **CATIA file name** ***Nomenclature*** **CATIA nomenclature** ***Revision*** **CATIA revision** ***Definition*** **CATIA definition** ***Description*** **CATIA description**

The default implementation gets the instance properties for the given naming schema.

Name:	**CMIRIICustomGetExportInstanceProperties**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetExportInstanceProperties**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**IPdmNameValueSet getExportInstanceProperties (** **String namingSchema,** **IPdmNameValueSet properties,** **IPdmStatus statusObject);**
Description:	**This interface function is called to retrieve the reference properties for the given naming schema..** **namingSchema** **the identifier of the used naming schema** **properties** **the CATIA and the external properties. The following names are supported:** ***RelObid*** **relation ID**

	MatrixIndex **matrix index for multi quantity relation** ***CatInstanceName*** **instance name in CATIA** ***ExtInstanceName*** **instance name of mapping file (optional)** **statusObject** **the messages displayed in CATIA** **Returns the values for the reference (IPdmNameValueSet). The following name is supported:** *InstanceName* CATIA instance name

Script file customization

The following customization extension point is called while getting the script dataset file in Teamcenter.

The default implementation gets the script dataset based on a query dialog. A selection dialog with the found datasets is shown to the user.

Name:	**CMIRIICustomGetScript**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetScript**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**Object getScript (IPdmId scriptId,** **IPdmId masterId,** **String scriptDescription,** **String scriptFilename);**
Description:	**This interface function is called to retrieve the script dataset.** **scriptId** **the identifier of the selected script** **masterId** **the identifier of the master object** **scriptDescription** **the description of the selected script** **scriptFilename** **the filename of the selected script** **Returns the script dataset (TCComponentDataset).**

Product Bounding Boxes

With Configurable Node Behavior in CMI it is possible that CATProducts reference 3D geometry that is not stored in Teamcenter, but rather included from the CATIA environment, e.g. to include Standard Parts from catalogs.

As these parts are not represented in Teamcenter, they would not be part of the result of a DMU neigbourhood search. This is adressed by the CATProduct bounding boxes, which are placeholders for exactly those Parts that are external to Teamcenter.

DMU neighourhood search will expand assemblies whose products have an eligible bounding box, in addition to those assemblies that contain eligible CATParts. In particular, this is useful if you create assembly or product JT files, as it will ensure that these files are visualized when appropriate.

Configuration

Please set the Teamcenter preference CMIWorkWithProductBBox=1 in order to enable the support for bounding box at the product datasets. (This bounding box represents the configurable ignored children in CATIA.)

In the CATIA environment set CMI_CALC_BBOX_FOR_IGNOREDCHILDREN=ON to enable calculation of the bounding box when a modified CATProduct is updated.

Set CMI_CALC_BBOX_FOR_IGNOREDCHILDREN=FORCE in a migration scenario for existing data. This will enable the save of bounding boxes whenever a CATProduct is writeable (to provide existing products with a bounding box, where the external parts were already present).

A bounding box will only be stored if CATParts or other geometry files are ignored due to a configurable embedded node behavior, and the new <BBox> tag is configured in the CMI configuration file (see *Configurable Behaviors in CATIA V5*)

Example:

```
<ConfigurableBehaviors>
    <ConfigurableBehavior UniqueID = "EmbeddedNode_STD_Ignore">
        <BehaviorType>EmbeddedNodeBehavior</BehaviorType>
        <PartNumberPrefix>STD_</PartNumberPrefix>
        <Behavior>IgnoreNode</Behavior>
        <BBox>true</BBox>
    </ConfigurableBehavior
</ConfigurableBehaviors>
```

In this example, CATIA component nodes with a part number beginning with STD_ are ignored by CMI. Any CATPart files below this node will not be stored in Teamcenter, but will be pulled from the environment.

During Update or Synchronize the bounding boxes of these CATParts will be combined into a single enclosing bounding box and will be stored in the CATIA Product dataset in Teamcenter.

Display CATIA Node Name in Synchronize

The width and visibility of columns in the Synchronize dialog (see Figure 9) can be configured by editing the file

`MACHINETYPE\resources\msgcatalog\CMIUpdateCreateDialog.CATRsc`

in the CMI CATIA installation dirctory (MACHINETYPE can be intel_a, win_b64, etc).

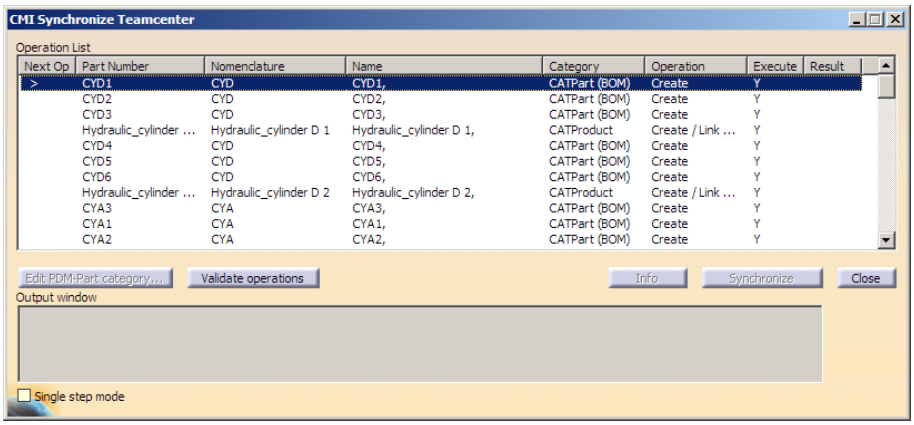

Figure 9: CMI Synchronize Teamcenter Dialog

E.g. the column *Name* is not shown by default, but it contains the text configured in the customized display for Reference Product (see Figure 10).

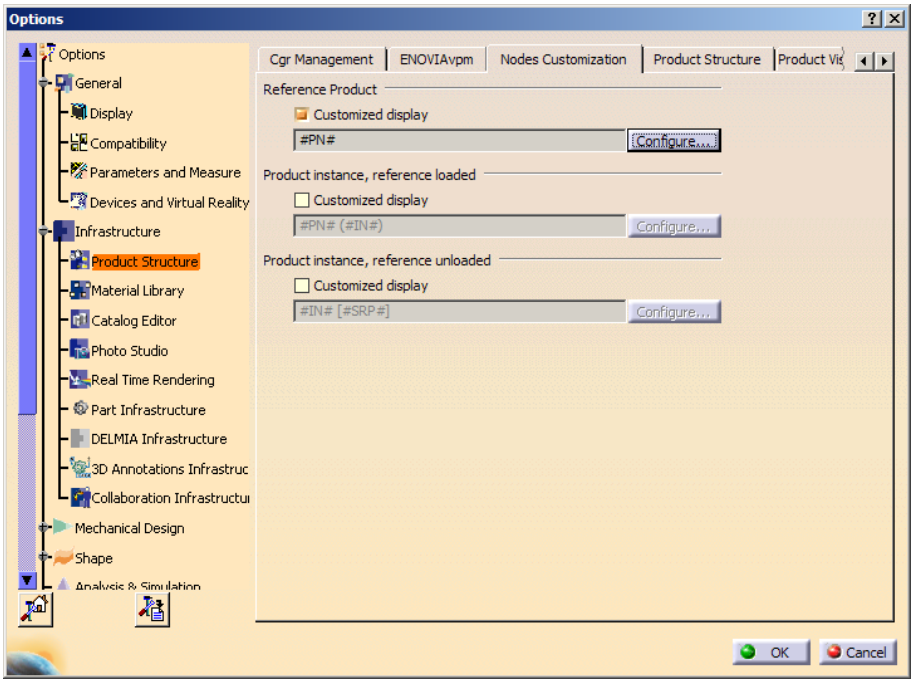

Figure 10: CATIA Node Customization options

To make the column *Name* visible by default, the following setting has to be changed in the file **`CMIUpdateCreateDialog.CATRsc:`**

```
// Column width of Name
JobMultiList.ColumnWidth4 = "0";
```

To hide `Nomenclature` and show `Name` instead you have to set:

```
// Column width of Nomenclature
JobMultiList.ColumnWidth3 = "0";
// Column width of Name
JobMultiList.ColumnWidth4 = "16";
```

Model Type customization

The following customization extension point is called to get the Model Type of a dataset from Teamcenter.

The default implementation gets the Model Type from the TypeInfoForm attached to the Dataset.

Name:	**CMIRIICustomGetModelType**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomGetModelType**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**String getModelType (TCComponentItemRevision revision, TCComponentDataset dataset, IPdmStatus statusObject);**
Description:	**This interface function is called to retrieve the model type from the dataset.** **revision** **the identifier of the referenced ItemRevision object** **dataset** **the identifier of the dataset object** **statusObject** **the messages displayed in CATIA** **Returns the model type.**

The following customization extension point is called to set the Model Type infos to the dataset in Teamcenter.

The default implementation gets the Model Type from the TypeInfoForm attached to the Dataset.

Name:	**CMIRIICustomSetModelType**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomSetModelType**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void setModelType (TCComponentItemRevision revision, TCComponentDataset dataset, String modelType, IPdmStatus statusObject);**
Description:	**This interface function is called to set the model type to the dataset.** **revision** **the identifier of the referenced ItemRevision object**

	dataset **the identifier of the dataset object** **modelType** **the model type to set** **statusObject** **the messages displayed in CATIA** **Returns the model type.**

```
JobMultiList.ColumnWidth4 = "16";
```

Item Revise customization

The following customization extension point is called to revise an ItemRevision in Teamcenter.

The default implementation revises an ItemRevision in Teamcenter.

Name:	**CMIRIICustomReviseItem**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomReviseItem**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**TCComponentItemRevision reviseItem(** **TCComponentItemRevision itemRevision,** **IPdmStatus statusObject);**
Description:	**This interface function is called to revise an ItemRevision.** **itemRevision** **the identifier of the referenced ItemRevision object** **statusObject** **the messages displayed in CATIA** **Returns the revised item revision.**

The following customization extension point is called after the reviseItem custom point.

The default implementation renames the CATIA Datasets with the new Revision ID if needed.

Name:	**CMIRIICustomReviseItemPost**
Plugin ID:	**com.tsystems.cmi.r2.interfaces**
Interface:	**ICustomReviseItemPost**
Package	**com.tsystems.cmi.r2.interfaces.custom**
Function:	**void reviseItemPost(** **Object origItemRevison,**

	Object revisedItemRevison, IPdmStatus statusObject);
Description:	**This interface function is called to set the model type to the dataset.** **origItemRevison the identifier of the referenced original ItemRevision object** **revisedItemRevison the identifier of the referenced revised ItemRevision object** **statusObject the messages displayed in CATIA**

CAA Customization

An API is provided for performing custom steps before and after the Synchronize function in CATIA, as callback functions in CAA. The callback is implemented by loading a customer compiled shared library (dll) that contains pre-specified static functions.

The following callback functions are available:

extern "C" HRESULT CMICusPrepareSynchronize (CATUnicodeString &sFeedback, CATBoolean &bCancel)

Called after Command is activated but before Synchronize dialog is filled.

`sFeedback`	String can be set and will be piped through to consecutive customizing messages, eg. to suspend the custom actions.
`bCancel`	Return `bCancel==true` to cancel the Command. Customization is responsible for user message.
`HRESULT`	`S_OK` if success `E_FAIL` if failure Failure does not stop processing.

extern "C" HRESULT CMICusPreSynchronize (CATUnicodeString &sFeedback, CATBoolean &bCancel)

Called after Synchronize button is pushed.

`sFeedback`	String is supplied and will be piped through to consecutive customizing messages, eg. to suspend the custom actions.
`bCancel`	Return `bCancel==true` to cancel the action. Customization is responsible for user message. Synchronize button will be in deactivated state.

HRESULT	S_OK if success E_FAIL if failure Failure does not stop processing.

extern "C" HRESULT CMICusPostSynchronize (const CATUnicodeString &sFeedback, const CATBoolean bCompleted)

Called after Synchronize button is pushed.

sFeedback	String is supplied.
bCompleted	TRUE if Synchronize succeeded
HRESULT	S_OK if success E_FAIL if failure Failure does not stop processing.

A sample Visual Studio Project to create the customizing DLL is provided in data\CMICAA\CMICusCallbackWorkspace.zip

Configuration

The custom callbacks are enabled by setting

CMI_ENABLE_CUSTOMIZATION=ON in your CATIA environment.

CMICusCallback.dll is installed in the %PATH%

CHAPTER 5

Teamcenter configuration variables

CMI Teamcenter Preferences

These are the possible settings for CMI within the Teamcenter preferences.

CMI Properties for the different datasets, DatasetRelation, DatasetType, and NamedReference

CMIArchiveDatasetRelation	The type of the relation between the item revision and archive dataset. Default value is "IMAN_specification".
CMIArchiveDatasetType	The type of the dataset containing the archive file. Default value is "CMI2Archive".
CMIArchiveNamedReference	Named reference in the dataset containing the archive file. Default value is "CMIArchive".
CMIAuxCgrDatasetRelation	The type of the relation between the item revision and the dataset containing the auxiliary cgr file. Default value is "IMAN_specification".
CMIAuxCgrDatasetType	The type of the dataset containing the auxiliary cgr file. Default value is "CMI2AuxCgr".
CMIAuxModelDatasetRelation	The type of the relation between the item revision and the dataset containing the auxiliary model file. Default value is "IMAN_specification".
CMIAuxModelDatasetType	The type of the dataset containing the auxiliary model file. Default value is "CMI2AuxModel".
CMIAuxPartDatasetRelation	The type of the relation between the item revision and the dataset containing the auxiliary part file. Default value is "IMAN_specification".
CMIAuxPartDatasetType	The type of the dataset containing the auxiliary part file. Default value is "CMI2AuxPart".
CMIAuxRepDatasetRelation	The type of the relation between the item revision and the dataset containing the auxiliary representation file. Default value is "IMAN_specification".
CMIAuxRepDatasetType	The type of the dataset containing the auxiliary representation file. Default value is "CMI2AuxRep".
CMIAuxAnalysisDatasetRelation	The type of the relation between the item revision and the dataset containing the auxiliary analysis file. Default value is "IMAN_specification".

CMIAuxAnalysisDatasetType	The type of the dataset containing the auxiliary analysis file. Default value is "CMI2AuxAnalysis".
CMIAnalysisNamedReference	The named reference in the dataset containing the analysis file. Default value is "CATAnalysis".
CMICacheCgrDatasetRelation	The type of the relation between the item revision and the dataset containing the cache cgr file. Default value is "IMAN_specification".
CMICacheCgrDatasetType	The type of the dataset containing the cache cgr file. Default value is "CMI2CacheCgr".
CMICacheCgrNamedReference	The named reference in the dataset containing the cache cgr file. Default value is "cgr".
CMICatalogDatasetType	The type of the dataset containing the catalog file. Default value is "CMI2Catalog".
CMICatalogNamedReference	The named reference in the dataset containing the catalog file. Default value is "catalog".
CMICgmDatasetRelation	The type of the relation between the item revision and the dataset containing the cgm file. Default value is "IMAN_specification".
CMICgmDatasetType	The type of the dataset containing the cgm file. Default value is "CMI2Cgm".
CMICgmNamedReference	The named reference in the dataset containing the cgm file. Default value is "cgm".
CMICgrDatasetRelation	The type of the relation between the item revision and the dataset containing the cgr file. Default value is "IMAN_specification".
CMICgrDatasetType	The type of the dataset containing the cgr file. Default value is "CMI2Cgr".
CMICgrNamedReference	The named reference in the dataset containing the cgr file. Default value is "cgr".
CMIDerivedModelDatasetRelation	The type of the relation between the item revision and the derived model dataset. Default value is "IMAN_specification".
CMIDerivedModelDatasetType	The type of the dataset containing the derived model file. Default value is "CMI2DerivedModel".
CMIDesignTableDatasetRelation	The type of the relation between the item revision and the dataset containing the design table. Default value is "IMAN_specification".
CMIDesignTableDatasetType	The type of the dataset containing the design table file. Default value is "CMI2DesignTable".
CMIExcelNamedReference	The named reference in the dataset containing the Excel design table file. Default value is "xls".
CMITextNamedReference	The named reference in the dataset containing the text design table file. Default value is "txt".
CMIDrawingDatasetRelation	The type of the relation between the item revision and the dataset containing the drawing file. Default value is "IMAN_specification".

CMIDrawingDatasetType	The type of the dataset containing the drawing file (e.g. CATDrawing). Default value is "CMI2Drawing".
CMIDrawingNamedReference	The named reference in the dataset containing the drawing file. Default value is "CATDrawing".
CMIMappingDatasetRelation	The type of the relation between the item revision and the mapping dataset. Default value is "IMAN_specification".
CMIMappingDatasetType	The type of the dataset containing the mapping file. Default value is "CMI2Mapping".
CMIMappingNamedReference	The named reference in the dataset containing the mapping file. Default value is "xml".
CMIModelDatasetRelation	The type of the relation between the item revision and the dataset containing the model file. Default value is "IMAN_specification".
CMIModelDatasetType	The type of the dataset containing the model file. Default value is "CMI2Model".
CMIModelNamedReference	The named reference in the dataset containing the model file. Default value is "model".
CMIModOnPartDatasetRelation	The type of the relation between the item revision and the ModOn part dataset. Default value is "IMAN_specification".
CMIModOnPartDatasetType	The type of the dataset containing the ModOn part file (e.g. CATPart). Default value is "CMI2ModOnPart".
CMIModOnProductDatasetRelation	The type of the relation between the item revision and the ModOn product dataset. Default value is "IMAN_specification".
CMIModOnProductDatasetType	The type of the dataset containing the ModOn product file (e.g. CATProduct). Default value is "CMI2ModOnProduct".
CMIPartDatasetRelation	The type of the relation between the item revision and the dataset containing the part file. Default value is "IMAN_specification".
CMIPartDatasetType	The type of the dataset containing the part file (e.g. CATPart, model, cgr). Default value is "CMI2Part".
CMIPartNamedReference	The named reference in the dataset containing the part file. Default value is "CATPart".
CMIPdfDatasetRelation	The type of the relation between the item revision and the dataset containing the pdf file. Default value is "IMAN_specification".
CMIPdfDatasetType	The type of the dataset containing the pdf file. Default value is "PDF" (Teamcenter standard).
CMIPdfNamedReference	The named reference in the dataset containing the pdf file. Default value is "PDF_Reference" (Teamcenter standard).
CMIProcessDatasetRelation	The type of the relation between the item revision and the dataset containing the process file. Default value is "IMAN_specification".

CMIProcessDatasetType	The type of the dataset containing the process file (e.g. CATProcess). Default value is "CMI2Process".
CMIProcessNamedReference	The named reference in the dataset containing the process file. Default value is "CATProcess".
CMIProductDatasetRelation	The type of the relation between the item revision and the dataset containing the product file. Default value is "IMAN_specification".
CMIProductDatasetType	The type of the dataset containing the product file (e.g. CATProduct). Default value is "CMI2Product".
CMIProductNamedReference	Named reference in the dataset containing the product file. Default value is "CATProduct".
CMIRepDatasetRelation	The type of the relation between the item revision and the dataset containing the representation file. Default value is "IMAN_specification".
CMIRepDatasetType	The type of the dataset containing the representation file. Default value is "CMI2Rep".
CMIRepresentationNamedReference	The named reference in the dataset containing the representation file. Default value is "Representation".
CMIScriptDatasetType	The type of the dataset containing the script file (e.g. CATScript). Default value is "CMI2Script".
CMIScriptNamedReference	The named reference in the dataset containing the script file. Default value is "CATScript".
CMITifDatasetRelation	The type of the relation between the item revision and the dataset containing the tif file. Default value is "IMAN_specification".
CMITifDatasetType	The type of the dataset containing the tif file. Default value is "TIF" (Teamcenter standard).
CMITifNamedReference	The named reference in the dataset containing the tif file. Default value is "TIF_Reference" (Teamcenter standard).
CMIDependendDatasetRelation	The type of the relation for the dependency relation between datasets. Default value is "IMAN_external_object_link".
CMICadAttrFormRelation	The type of the relation between dataset and CMICadAttrForm. Default value is "IMAN_external_object_link".
CMICatiaInfoFormRelation	The type of the relation between dataset and CMICatiaInfoForm. Default value is "IMAN_external_object_link".

CMI Properties for PartType and ProductType

CMIPartType	The item type used for the CATIA part representation. Default value is "Item".
CMIProductType	The item type used for the CATIA product representation. Default value is "Item".

CMI Properties for Create of Item and Dataset

CMICreateItemInteractive	If set to 1 the create dialog for the item will be displayed. Default value is "1".

CMICreateItemDialog_*className*_Height	Customizes the height of the input panel of the create item dialog for the class *className*.
CMICreateItemDialog_*className*_Width	Customizes the width of the input panel of the create item dialog for the class *className*.
CMICreateItemItemFormList	List containing the type of the forms to be used in the create item dialog. It can be defined for which item type the form should be created. The item type has to be added with a “@”. If the item type is not given, then the form will be used for every item. It can be defined with which relation type the form should be related to the item. The relation type has to be added with a ":". Otherwise the default relation type will be used. (e.g. “Item Master”, “Item Master:IMAN_reference”, “Item Master@Item:IMAN_reference”) Default value is "Item Master@Item:IMAN_reference".
CMICreateItemItemRevisionFormList	List containing the type of the forms to be used in the create item revision dialog. It can be defnied for which item revision type the form should be created. The item revision type has to be added with a “@”. If the item revision type is not given, then the form will be used for every item revision. It can be defined with which relation type the form should be related to the item revision. The relation type has to be added with a ":". Otherwise the default relation type will be used. (e.g. “ItemRevision Master”, “ItemRevision Master:IMAN_reference” , “ItemRevision Master@ItemRevision:IMAN_reference”) Default value is "ItemRevision Master@ItemRevision:IMAN_reference".
CMICreateItemItemTypeList	The list of the valid item types for the create item dialog. Default value is {"Item"}.
CMICreateItemDialogRememberAttributes	If set to 1 the attributes in the create item dialog will be stored and displayed in the new create dialog for the same class/item type. Default value is "0".
CMISelectItemType	If set to 1 the select item type functionality is switched on. Default value is "0".
CMIUsePartNumberId	If set to 1 try to use the part number as ID for the new assemblies. If it exists an autogenerated number is used. Default value is "0".
CMILinkNewItemToNewstuffFolder	If set to 1 link the newly created items to the Newstuff folder. Default value is "1".
CMICreateDatasetInteractive	If set to 1 the create dialog for the dataset will be displayed. Default value is "0".
CMICreateDatasetInteractiveExclusionList	The list of the dataset types which do not use the interactive create dataset dialog. Default value is {"CMI2Part", “CMI2Product”, “CMI2CacheCgr”, “PDF”, “TIF”}.
CMICreateDatasetDialog_*className*_Height	Customizes the height of the input panel of the create dataset dialog for the class *className*.
CMICreateDatasetDialog_*className*_Width	Customizes the width of the input panel of the create dataset dialog for the class *className*.
CMIDatasetNameEditable	If set to 1 the dataset name in the create dataset dialog can be edited. Default value is "0".

CMICreateDatasetDialogRememberAttributes	If set to 1 the attributes in the create dataset dialog will be stored and displayed in the new create dialog for the same class/dataset type. Default value is "0".
CMILinkNewDatasetToNewstuffFolder	If set to 1 link the newly created datasets to the Newstuff folder. Default value is "0".
CMICreateDialogRememberAttributes	If set to 1 the attributes in the create dialog will be stored and displayed in the new create dialog for the same class/dataset type. Default value is "0".
CMICreateDialog_*className*_Height	Customizes the height of the input panel of the Create Dialog for the class *className*. Default value is “100”.
CMICreateDialog_*className*_Width	Customizes the width of the input panel of the Create Dialog for the class *className*. Default value is “700”.

CMI Properties for Drawing

CMICreateItemForDrawing	If set to 1 when no item is referenced by the drawing it will be created. Default value is "1".
CMILinkDrawingToPart	If set to 1 link the newly created dataset with the CATDrawing object to the item object. Default value is "1".
CMIDrawingFileFormatsList	List of the file formats for the derived files of the CATDrawing. Supported values are "pdf" and "tif".

Partnumber handling for Aux (NoBOM) Files

CMIUseCatiaPartnumberForAux	If set to 1 use the part number from CATIA as part number of the auxiliary file (CMI2AuxPart, etc.) in Teamcenter. Default value is "0".
CMIUseCatiaPartnumberForAuxDatasetName	If set to 1 use the part number from CATIA as Dataset Name of the auxiliary file (CMI2AuxPart, etc.) in Teamcenter. Default value is "0".

CMI Properties for Filename

CMIUseCatiaFilenameForArchive	If set to 1 use the file name from CATIA as file name of the archive file in Teamcenter. Default value is "0".
CMIUseCatiaFilenameForAuxAnalysis	If set to 1 use the file name from CATIA as file name of the analysis file in Teamcenter. Default value is "0".
CMIUseCatiaFilenameForAux	If set to 1 use the file name from CATIA as file name of the aux file in Teamcenter. Default value is "0".
CMIUseCatiaFilenameForDesignTable	If set to 1 use the file name from CATIA as file name of the design table file in Teamcenter. Default value is "0".
CMIUseCatiaFilenameForDrawing	If set to 1 use the file name from CATIA as file name of the drawing in Teamcenter. Default value is "0".
CMIUseCatiaFilenameForMain	If set to 1 use the file name from CATIA as file name of the main file in Teamcenter. Default value is "0".

CMI Properties for Synchronize Dialog

CMISynchronizeDatasetDialog_Height	Customizes the height of the container of the Synchronize Dialog for the datasets. Default value is "60".
CMISynchronizeDatasetDialog_Width	Customizes the width of the container of the Synchronize Dialog for the datasets. Default value is "700".
CMISynchronizeItemDialog_Height	Customizes the height of the container of the Synchronize Dialog for the items. Default value is "60".
CMISynchronizeItemDialog_Width	Customizes the width of the container of the Synchronize Dialog for the items. Default value is "700".

CMI Properties for Edit → Options → CMI RII

CMISendCacheCgrToCatia	Send cache cgr files to CATIA. Accepted values: 0 - no, 1 - cache cgr only, 2 - CATPart and cache cgr. Default value is "2".
CMISendCgmToCatia	If set to 1 cgm datasets will be sent to CATIA during the "Send to CATIA" action. Default value is "1".
CMISendDrawingToCatia	If set to 1 drawing datasets will be sent to CATIA during the "Send to CATIA" action. Default value is "1".
CMISendVisuToCatia	If set to 1 visualization datasets will be sent to CATIA during the "Send to CATIA" action. Default value is "0".
CMISendAnalysisToCatia	If set to 1 analysis datasets will be sent to CATIA during the "Send to CATIA" action. Default value is "0".
CMIWorkWithCatia	Preference to work with which CAD System. Accepted values: V5 - CATIA V5, V4 - CATIA V4. Default value is "V5".
CMIReadMode	Preference for the read mode. Accepted values: STANDARD - Send the visible (expanded) children to CATIA, DMU - Send the DMU marked (checked) children to CATIA, and AUTO - Use DMURead Mode if the Viewer is active. Use Standard Read Mode if the Viewer is not active. Default value is “STANDARD”.
CMINewstuffItems	Preference for the newstuff settings for Items. Accepted values are STANDARD, INDIVIDUAL, and NONE. Default value is "STANDARD".
CMINewstuffItemsIndividualName	Preference for the newstuff settings for Items. The name of the individual newstuff folder is stored.
CMINewstuffItemsIndividualObid	Preference for the newstuff settings for Items. The object id of the individual newstuff folder is stored.
CMINewstuffDatasets	Preference for the newstuff settings for Datasets. Accepted values are STANDARD, INDIVIDUAL, and NONE. Default value is "NONE".
CMINewstuffDatasetsIndividualName	Preference for the newstuff settings for Datasets. The name of the individual newstuff folder is stored.

CMINewstuffDatasetsIndividualObid	Preference for the newstuff settings for Datasets. The object id of the individual newstuff folder is stored.
CMIDmuClearance	Preference for the DMU clearance in millimeter. Default value is "1.0".
CMIAltRepList	Contains a list of alternate representations. The first list object is the default (e.g. MASTER). When this list is the empty list then the Alternate Representation functionality is removed from the dialog.
CMIActualAltRepList	Contains the actual selected alternate representation ordered list. Must be in the list of alt reps (see CMI_ALT_REP_LIST). Default value is {"MASTER"}.
CMIUseAltRepFilter	If set to 1 the Alternate Representation Filter should be used. Default value is "0".

CMI Properties for dataset handling, Read Write access for files in CATIA

CMIUseChangeableMeansOwner	If set to 1 the checkout state defines the changeable flag (Write Access in CATIA), changeable is true if the object is checked out to session user, otherwise false. If set to 0 the access rules of Teamcenter defines the changeable flag (Write Access in CATIA), changeable is true if the session user has modify access, otherwise false. Default value is "0".
CMIDatasetCheckOutAfterCreate	If set to 1 the newly created dataset is checked out by the current session user. Default value is "0".
CMIDatasetCheckOutRequired	If set to 1 the update of files requires that the dataset is checked out by the current session user. If the dataset is checked in and the dataset is not changed by another user, the dataset will be automatically checked out by the session user Default value is "0".

CMI Properties for Revise of Released objects, Item and Dataset

CMIItemRevisionReleasedStatusList	The list of the released status values for an item revision. Default value is {"TCM Released"}.
CMIDatasetReleasedStatusList	The list of the released status values for a dataset. Default value is {"TCM Released"}.

CMI Properties for the naming of Dataset revision attribute

CMIDatasetRevisionCarryOver	The value of the carry over for the dataset revision. Default value is "A".
CMIDatasetRevisionValues	The values of the dataset revision in ascending order. The CMI_DATASET_REVISION_CARRY_OVER will be used when the end of the list is reached. Default value is "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

CMI Properties for MasterForm and RevisionMasterForm in the create use case

CMIShowMasterFormProperties	If set to 1 show the dialog while creating the item in order to edit properties of the Master Form. Default value is "0".
CMIShowRevisionMasterFormProperties	If set to 1 show the dialog while creating the item in order to edit properties of the Revision Master Form. Default value is "0".

CMI Properties for the columns of the tree table in the CMI RII application

CMITreeTableColumnsShownPref	Columns to be displayed in the CMI tree table.
CMITreeTableShownColumnWidthsPref	Column widths for the CMI tree table.
CMIPrefPreloadProperties	If set to 1 properties are preloaded when creating lines in the CMI RII Application, even if the line is not visible in the window. Default value is “0”.

CMI Properties for triggering the translation service

CMITriggerTranslation	If set to 1 the translation service is triggered for updated CATIA files. Default value is "0".
CMI_*className*_Priority	The preferences for the priority of the translation service which will be triggered for the CATIA file dataset of class *className*.
CMI_*className*_Provider	The preferences for the provider of the translation service which will be triggered for the CATIA file dataset of class *className*.
CMI_*className*_Translator	The preferences for the translator of the translation service which will be triggered for the CATIA file dataset of class *className*.
CMI_*className*_Priority_2	The preferences for the priority of the second translation service which will be triggered for the CATIA file dataset of class *className*. Analog for the next translation services from 3.
CMI_*className*_Provider_2	The preferences for the provider of the second translation service which will be triggered for the CATIA file dataset of class *className*. Analog for the next translation services from 3.
CMI_*className*_Translator_2	The preferences for the translator of the second translation service which will be triggered for the CATIA file dataset of class *className*. Analog for the next translation services from 3.

miscellaneous

CMIWorkWithDesignTables	If set to 1 the CMI Design Table support is enabled. Default value is "0".
CMIUseCurrAppForRead	Try to use the current application for reading, if the application is supported (e.g. Structure Manager). Default value is "1".
CMIEnableUserDefProps	If set to 1 the support for user defined properties will be enabled. Default value is "0".
CMIRepresentationTypeList	List of file types for representations. Default value is {}.
CMIDependentTypeList	List of supported MML link types. Default value is {"CCP","Design"}.
CMIStoreCatiaInfos	If set to 1 store CATIA Information (e.g. CATIA Release Version) in the CMI2CatiaInfoForm. Default value is “0”.
CMIValidateCatiaVersion	If set to 1 validate the CATIA Version Information (e.g. CATIA Relase Version) before download of file. Default value is "0".
CMIUseExistingFindnoForCreateRel	If set to 1 use the existing (for same PartNumber) FindNo in the assembly for new instances. Default value is "0".

CMIUseNamingSchemaDialog	If set to 1 use a dialog to ask the user for the naming schema to be used for the export. Otherwise no naming schema will be used for the export. Default value is "0".
CMITemplateFolderType	The type of the folder containing the templates. Default value is “CMI2TemplateFolder”.
CMICreateItemForCatalog	If set to 1 an item will be created for the catalog dataset and they will be related. Default value is "0".
CMIDoNotLoadZeroQuantity	If set to 1 the used parts with quantity = 0 will not be loaded in CATIA V5. Default value is "0".
CMISetQuantityForCreateBomRel	If set to 1 the quantity of the newly created BOM relation will be set to "1". Default value is "0".
CMIUseAskModelType	If set to 1 the ask model type panel is shown in the create dataset dialog. Default value is "0".
CMIWorkWithProductBBox	If set to 1 the support for bounding box at the product datasets is enabled. (This bounding box represents the configurable ignored children in CATIA.) Default value is "0".
CMIApplicationUseCustomIcons	If set to 1 the custom icons in CMI RII Application are used. Default value is “0” to use the class icons from Teamcenter.

CHAPTER 6

CMI CATIA V5 RII Installation Package Structure

Directories

Following figure shows the standard directory tree of the CMI CATIA V5 RII installation package.

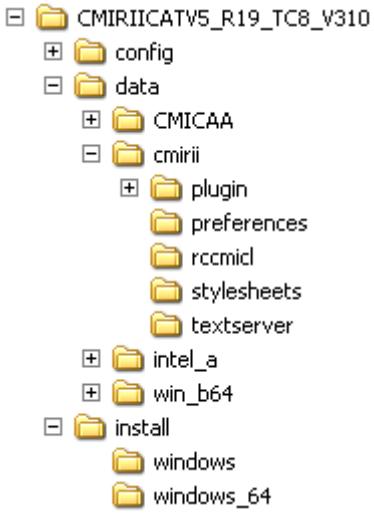

Figure 11: Directory structure of the CMI CATIA V5 RII installation package

The *config* directory contains sample CATIA V5 configuration file *defines.txt* for Windows. These environment settings will be used for the start script of CATIA V5.

The *data* directory contains the files for the Teamcenter Client and Server installation and the binary distributions for the CMICATV5 module for the supported operating system mnemonics.

The files for the Teamcenter Client and Server installation are stored as follows in the subdirectories of *cmirii*:

`plugin/`	Contains the directory of the ressource files and the Java jar files for the Teamcenter Rich Client.
`preferences/`	Contains the default preferences for Teamcenter Rich Client in two files: *cmi_preferences.xml* and *cmi_dialog_preferences.xml*.
`rccmicl/`	Contains the Java jar file for the communication between the Teamcenter Rich Client and CATIA V5.
`stylesheets/`	Contains the stylesheets for the forms.
`textserver/`	Contains texts that can be copied to the Teamcenter text server.

The supported operation systems and their mnemonics are:

Windows 32-Bit	intel_a
Windows 64-Bit	win_b64

The mnemonic “intel_a” will be chosen as an example of a CMICATV5 installation directory on Windows 32-Bit.

Following figure shows the directory tree of ”intel_a”.

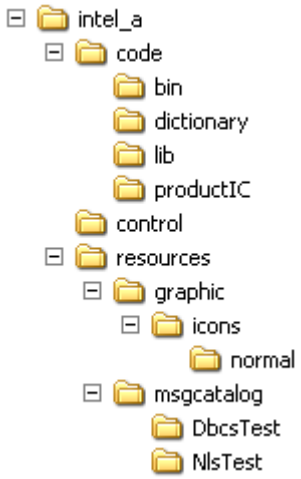

Figure 12: Directory structure of the CMICATV5 installation directory on Windows 32-Bit

`msgcatalog/` If you want to change text messages in CATIA V5 the files with the extension CATNls in the **`msgcatalog`** directory can be changed. This message files contain string variables. After adapting this strings changes will be shown in CATIA V5 (see examples below!).

All other files in the CMICATV5 installation directory should not be touched.

Files

This section describes some important files and their meaning. The files have a text message catalog with messages displayed in CATIA V5.

The following example shows the content of the file CMIUpdateCommandHeader.CATNls:

```
// (c) T-Systems 2001


//============================================================================
// Message catalog for the CMIAddinHeader command headers of the
// CMIAddin addin
//============================================================================


CMIUpdateCommandHeader.CMIUpdateCommandHeader.Category  = "CMI" ;
CMIUpdateCommandHeader.CMIUpdateCommandHeader.Title     = "Update" ;
```

```
CMIUpdateCommandHeader.CMIUpdateCommandHeader.ShortHelp = "Update Teamcenter" ;

CMIUpdateCommandHeader.CMIUpdateCommandHeader.Help      = "Update the active
window in Teamcenter" ;

CMIUpdateCommandHeader.CMIUpdateCommandHeader.LongHelp  = "Update

This command updates positions and files of the active window in Teamcenter." ;
```

If the mouse pointer is over the tool icon (in this example: the “Update Teamcenter” icon) the *Title* you will see in the status line before the command line. The *ShortHelp* messages will appears in the tooltip and the *Help* message appears in the status line left. After moving the “What’s This?” icon to the toolbar icon the text in *LongHelp* will be shown.

The following changeable files have the same structure as this example file.

resources/msgcatalog/CMIReadCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Read from Workbench”.

resources/msgcatalog/CMIUpdateCommandHeader.CATNls
This file contains the text nessages catalog for the CMI command

“Update Teamcenter”.

resources/msgcatalog/CMIUpdateCreateInteractiveCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Synchronize Teamcenter”.

resources/msgcatalog/CMIHighlightInWBCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Highlight in Teamcenter”.

resources/msgcatalog/CMIBrowseInTCCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Browse in Teamcenter”.

resources/msgcatalog/CMIAttachArchiveCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Create/Attach an Archive”.

resources/msgcatalog/CMIGetOrigGeoCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Get original geometry from Teamcenter”.

resources/msgcatalog/CMIUsePdmStructureCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Insert from Teamcenter”.

resources/msgcatalog/CMIReplacePdmStructureCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Replace from Teamcenter”.

resources/msgcatalog/CMIAddTempCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Add Temp from Teamcenter”.

resources/msgcatalog/CMICompareVersionCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Compare Version”.

resources/msgcatalog/CMIBuildVisuCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

"Build Visualization for Synchronize in Teamcenter".

resources/msgcatalog/CMIReconnectCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Reconnect with Teamcenter”.

resources/msgcatalog/CMICatalogCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Manage Catalogs”.

resources/msgcatalog/CMISaveLocalCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Save Session”.

resources/msgcatalog/CMIRestoreLocalCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Restore Session”.

resources/msgcatalog/CMIInfoCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“CMI-Info”.

resources/msgcatalog/CMICheckInOutCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Check In/Out in Teamcenter”.

For the optional commands:

resources/msgcatalog/CMISaveAsCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“SaveAs in Teamcenter”.

resources/msgcatalog/CMICatalogReadCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Read Catalog”.

resources/msgcatalog/CMICatalogInsertScriptCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Insert CATScript into Catalog”.

resources/msgcatalog/CMICatalogUpdCreCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Update or Create Catalog”.

resources/msgcatalog/CMIUpdatePositionCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Update Position”.

resources/msgcatalog/CMIUpdatePartCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

“Update Part”.

resources/msgcatalog/CMIRestorePositionCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

"Restore Position".

resources/msgcatalog/CMIReviseCommandHeader.CATNls

This file contains the text messages catalog for the CMI command

"Revise".

resources/msgcatalog/CMICatDuaReadCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

"Automatic CATDUA".

resources/msgcatalog/CMIExportCommandHeader.CATNls
This file contains the text messages catalog for the CMI command

"Export to Folder".

For working with text messages in different languages the files in the directory **resources/msgcatalog** should be copied in different subdirectories and adapted there.

Following figure shows an example of possible subdirectories under **msgcatalog.**

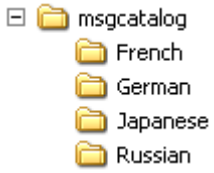

Figure 13: Example of directory structure of the CMICATV5 installation subdirectory msgcatalog

The English message files are located directly under the directory **msgcatalog.**

Customer dependent configurations for CATIA V5

Environment settings

Environment Variable	Comment
CMI_CREATE_V4_WITH_PARENT	If set the "Create V4" command only works in the Product Structure with a selected CATPart.
CMIXMAP	The location of the CATIA V5 Exchange Map.
CMI_DEBUG	If set to "ON" the debug output is written to stdout, else no output is created.
CMI_REMOVE_CMIREADCMD	If set to "ON" the "Read" command is not available.
CMI_REMOVE_CMIUPDATECMD	If set to "ON" the "Update" command is not available.
CMI_REMOVE_CMICREATECMD	If set to "ON" the "Create" command is not available.
CMI_REMOVE_CMISAVEASCMD	If set to "ON" the "SaveAs" command is not available.

CMI_REMOVE_CMIUPDATECREATECMD	If set to “ON” the “Synchronize” command is not available.
CMI_REMOVE_CMICREATEV4CMD	If set to “ON” the “Create V4” command is not available.
CMI_REMOVE_CMISAVEASV4CMD	If set to “ON” the “SaveAs V4” command is not available.
CMI_REMOVE_CMIINFOCMD	If set to “ON” the “Info” command is not available.
CMI_REMOVE_CMIHLINWBCMD	If set to “ON” the “Highlight in Workbench” command is not available.
CMI_REMOVE _CMIUPDATECREATEINTERACTIVECMD	If set to “ON” the “Update and Create Interactive” command is not available.
CMI_REMOVE_CMIINFOCMD	If set to “ON” the “CMI Info” command is not available.
CMI_SAVEAS_V4_WITH_DIRTYCHECK	If set to “ON” the “SaveAs V4” command only works with saved CATParts.
CMI_SAVEAS_V4_WITH_CMICHECK	If set to “ON” the “SaveAs V4” command only works with CATParts from CMI.
CMI_USE_COMMAND_SUBMENU	if set to "OFF" button similar-command grouping is turned off in the toolbar.
CMI_CALC_BBOX	Calculate and save bounding box info at CATParts on update (for DMU).
CMI_CALC_BBOX_FOR_IGNOREDCHILDREN	Set to "ON" to calculate the bounding box of configurable ignored children (with configuration BBOX=true) during update of the father product. Set to "FORCE" to update the bounding box whenever the father product is writeable.
CMI_DISABLE	Set to “ON” to get NO CMI toolbar. No check for CMI license is done.
CMI_ENABLE_VP_SAVE	Set to “1” to get the functionality “Save Virtual product”.
CMI_CALC_SHEET	When a drawing is saved, make the list of sheets available in Teamcenter.
CMI_REFERENCE_OF_DRAWING	When a drawing is saved, make the referenced Products/Parts available in Teamcenter.
CMI_BOUNDING_BOX_EXCLUDE_HIDDEN	Set to "ON" to exclude hidden objects from the bounding box calculation.
CMI_DISABLE_HIDESHOW	Set to "ON" to load the hide/show status as in the product (products not in the CMI Workbench are not hidden).
CMI_DISABLE_SET_TIMESTAMP	Set to "ON" to disable to set the last modification date to the files in the exchange map.
CMI_REMOVE_CMIADDTEMPCMD	Set to "ON" to disable the “AddTemp” command in CATIA V5.
CMI_ADDTEMP_PRAEFIX1	Default is "TMP" the first praefix for the rename of the Partnumbers and Filename for the AddTemp command.

CMI_ADDTEMP_PRAEFIX2	Default is "_" the second praefix for the rename of the Partnumbers and Filename for the AddTemp command The Default Praefix is set to “TMP#_”. # is a counter in CATIA.
CMI_ENABLE_CHECKMODTIMESTAMP	Default is "ON". Set to "OFF" to disable the fuctionality to check for "Saved" Files, which are saved by the native CATIA “Save” command; also, if set to "OFF" out of sync cgr files in the local cache are not treated by CMI.
CMI_ENABLE_CACHEMODE_RESETDOCLINKS	Default is "OFF". Set to "ON" if it should be attempted to reset (refresh) document links in Cache Mode. Default is OFF as R12 sp3 can't refresh geometric document links properly.
CMI_ENABLE_CMIUPDATEPOSITIONCMD	Set to "ON" to enable the “Update Position” command in CATIA V5.
CMI_ENABLE_CMIGETORIGGEOCMD	Set to "ON" to enable the “Get original Geometry” command in CATIA V5.
CMI_USERELEASEDCACHE	Set to "ON" to transfer CGR-files to the Released Cache.
CMI_RELEASEDCACHEDIR	Only used if CMI_USERELEASEDCACHE=ON. Sets the Released Cache dir used by CMI to a specific member of the list of Released Cache directories in CATIA. Default: not set -> CMI uses the first member of the list. If set to a member, this member must be part of the list.
CMI_CLEANRELEASEDCACHE	Only used if CMI_USERELEASEDCACHE=ON. Default is "OFF". Set to "ON" if you do not use version independent file names.
CMI_CREATETEMPCGRCOMP	Only used if CMI_USERELEASEDCACHE=ON. Default is "OFF". Set to "ON" to use temporary CATIA components which contain the related CGR as shape representation, instead of the original CATPart.
CMI_CHECK_LINKED_DRAWING	If set to "ON" then with Create/Create & Link/Save As the CATPart is checked for a related opened Drawing.
CMI_CHECK_LINKED_PRODUCT	If set to "ON" then with Create/Create & Link/Save As the CATPart is checked if it is opened in another product.
CMI_DEFAULT_UNIT	If no unit of measurement is passed from the PDM system to CATIA, then the unit may be set here.
CMI_CONFIGURATION_FILE	Full file name and path to the CMI XML Configuration file (alternative/complement to system environment variable declaration).
CMI_ENABLE_SINGLEPARTMODUS_READ	If set to "ON" the "Single Part Modus" option in the CMI Options CATIA V5 property page is enabled.
CMI_CONNECTPDM	String to use to override the default command-line omfcl call.
CMI_ENABLE_NATIVEPRODUCTTRAFO	Set "ON" to enable the option to suppress CMI Transformation.

CMI_ENABLE_APPLY_VISUMODE	If set to "ON" CMI will switch CATParts that it loads in Design mode, back to Visualization mode if possible (supported by CATIA beginning R13).
CMI_RESTORE_POSITION	"ON" / "OFF" --> if "ON" user can reset Matrix position to original CMI Matrix position within CATIA V5.
CMI_DRAWING_CHECKUPDATEOFASSEMBLY	If set to "ON" CMI will warn if you update a CATDrawing when at least one related CATPart or CATProduct is not saved yet.
CMI_ENABLE_UPDATEPOSITIONDIALOG	If set to "ON" then show a dialog of modified positions at update.
CMI_DISABLE_LOAD_STDCATPARTS	If set to "ON" then standard CATParts will not be loaded into design mode automatically. They are identified not by their part number (which is not available in cache mode) but by their instance name, which must be "Part Number + .(dot) + some string".
CMI_IGNORE_NONCMI_ROOT_CHILDREN	If set to "ON" and "Use Virtual Root" is turned on in the CMI Settings in CATIA, then any children attached to the virtual root that are not from CMI will be ignored during Update/Sync commands.
CMI_DISABLE_SAVETOXMAP	Set to "ON" if files located outside of the exchange map should NOT be moved into the exchange map before they are created or updated in the PDM system.
CMI_PACK_ARCHIVE_CMD	String to use to override the default command-line for packing/zipping archive files. Default is "zip -0 -q -j", where -0 is "store only", -q is "quiet operation", -j is " junk (don't record) directory names".
CMI_UNPACK_ARCHIVE_CMD	String to use to override the default command-line for unpacking/unzipping archive files. Default is "unzip -o -j -q -d", where -o is "Override without prompting", -j is "do not use Directory names", -q is "quiet mode", -d <Dir> "extract to dir".
CMI_REMOVE_CMIATTARCCMD	If set to “ON” the “Attach Archive” command is unavailable.
CMI_GLOBAL_DISABLE	If set to "ON" the “CMI General Update Addin” commands are disabled.
CMI_DISABLE_REPLACE_WRONG_PRD	If set to "ON" the following functionality is disabled: During a Read, if CMI recognizes that a file with a different UUID has been received from the PDM system instead of the file UUID named in the parent CATProduct, then the new file is attached in place of the old.
CMI_ENABLE_VALIDATE_BEFORE_UPD	Set to "ON" to enable customer specific validation of all operations in “Synchronize” command before a sync may be executed.
CMI_DISABLE_STEP_SYNC	Set to "ON" to remove the "Synchronize operations singly" check box from the “Synchronize” dialog.
CMI_DISABLE_NEW_CGR	If set to "ON" this disallows the addition of new CGR files to the product structure.
CMI_DISABLE_NEW_CGR_INSTANCE	If set to "ON" this disallows the addition of new instances of CGR files to the product structure.

CMI_DISABLE_NEW_V4MODEL	If set to "ON" this disallows the addition of new V4 Model files to the product structure.
CMI_DISABLE_NEW_V4MODEL_INSTANCE	If set to "ON" this disallows the addition of new instances of V4 Model files to the product structure.
CMI_ENABLE_ARCHIVE_VALIDATION	If set to "ON" the Archive Validation message in Teamcenter is called before Archive Create/Update.
CMI_DISABLE_ANALYSIS_IN_ARCHIVE	If set to "ON" the support for CATAnalysis in CMIArchives is disabled.
CMI_ENABLE_ANALYSIS_COMPUTATION_IGNORE	If set to "ON" the computations in CATAnalysis is ignored, else the computations must be deleted by the user.
CMI_DISABLE_MODEL_IN_ARCHIVE	If set to "ON" the support for V4 models in CMIArchives is disabled.
CMI_DISABLE_CGR_IN_ARCHIVE	If set to "ON" the support for cgr in CMIArchives is disabled.
CMI_ENABLE_ARCHIVE_ROOT_PRODUCT_ONLY	If set to "ON" the Root in CMIArchives must be a CATProduct.
CMI_REMOVE_CMIMODNCARCCMD	If set to "ON" the “Modify non CATIA” command is not available.
CMI_ENABLE_UPD_MODELSELECT_DIALOG	If set to "ON" a dialog for model update selection is shown in the “Update” command.
CMI_ENABLE_CHECKISUPTODATE	If set to "ON" the user can cancel the Update/Create/ Save As action if a CATPart or CATProduct is not synchronized in the CATIA Session.
CMI_ENABLE_CMIEXTERNALDOCCMD	If set to "ON" the command to load referenced documents is available. Additional software is required for this.
CMI_DISABLE_LOADOK_MESSAGE	If set to "ON" no message is shown after a successful Read from Teamcenter.
CMI_DISABLE_UPDATE_WB	If set to "ON" the Workbench is not updated with new items created during Synchronize.
CMI_ENABLE_CHECKISUPTODATE	If set to "ON" a check is performed during Update if any geometry needs to be updated in CATIA.
CMI_ENABLE_CHECKMULTIEMBARC	If set to "ON" ambiguous (same part number) local components are disallowed in an archive.
CMI_GETPOINTEDDOCUMENTS	if set to "ON" dependencies based on referenced documents are created in Teamcenter. This requires additional software.
CMI_ENABLE_CHECK_PRD_VAL_IGNORE	If set to "ON" the validate function of Synchronize succeeds if a product with no structural changes is modified and has to be updated.
CMI_ENABLE_CMIOPTIONSCMD	If set to "ON" the “CMI Options” dialog command button is available.
CMI_ENABLE_STDPARTINFO	If set to "ON" the standard Part infos are requested from Teamcenter, needed for the Component standard part integration.
CMI_REMOVE_CMIUSEPDMSTRUCTURECMD	If set to "ON" the "Insert from Teamcenter" command is not available.
CMI_REMOVE_CMIREPLACEPDMSTRUCTURE	If set to "ON" the "Replace from Teamcenter" command

CMD	is not available.
CMI_REPLACE_ALLOW_NONBOM	Set to "ON" to allow to replace NonBom geometries with "Replace from Teamcenter".
CMI_ENABLE_RESETINVALIDPOSCMD	If set to "ON" the "Reset Invalid Position" command is available.
CMI_ENABLE_CHECK_INVALID_POS	If set to "ON" during Update all models are checked for invalid positions.
CMI_BOM_PART_DEFAULT_FOR_SYNC	Set the default value for new Models in the sync dialog: NOT_SET, BOM, all ather settings are normal models.
CMI_ENABLE_ASK_FOR_BOM_PART	If set to "ON" the user is asked which kind of Part should be created in Teamcenter.
CMI_ENABLE_CMIARCHIVE_CREATE	If set to "ON" it is possible to create (Attach) archives without parent and without CMI Parent.
CMI_CAT_ENV_SCRIPT	Points to the cat start file (full path with extension) which is used to start the CMISender executable (Used in Teamcenter and CNEXT environment), CNEXT startup will create the file for the actual CATIA and the Teamcenter Rich Client will use this for starting the CMISender.
CMI_DISABLE_CMIBUILDVISUCMD	If set to "ON" the “CMI BuildVisu” command is not available.
CMI_DISABLE_CREATE_CATDRAWING	If set to "ON" the creation of new CATDrawings is disabled.
CMI_DISABLE_CREATE_CATPART	If set to "ON" the creation of new CATParts without parents is disabled.
CMI_DISABLE_CMIRECONNECTCMD	If set to "ON" the Reconnect” command is not available.
CMI_ENABLE_ARCHIVE_CACHE	If set to "ON" the archive file can handle Release Cache files.
CMI_PDM_MANAGED_STDCATPARTS	If set to "ON" then CMI treats standard part geometry (CATPart files) like regular component part geometry, that is, the geometry is expected to be attached to a document describing the Component in PDM and it is transferred to the exchange map at "To Catia".
CMI_PACK_ADD_ARCHIVE_CMD	String to use to override the default command-line for Adding files to archives. Default is "cmi_zip -0 -q -j", where -0 is "store only", -q is "quiet operation", -j is " junk (don't record) directory names".
CMI_PACK_ARCHIVE_MAX_FILES	Define the maximal number of files which will be packed with the Zip command. If not set, Default is 100. Used to reduce the needed length of the command line for the system call.
CMI_NO_PUBLIC_POS_UPDATE	Do not try to update transformations under products that are read only.
CMI_PACK_ARCHIVE_MAX_COMMANDLINE	Define the maximum length (in Bytes) of the system call which will pack a CMIArchive. If not set use system limits of the OS. If the limit is reached, the pack Archive command will be split (see CMI_PACK_ADD_ARCHIVE_CMD).

CMI_ARCHIVE_AUTOUPDATE	Automatic update after "Attach an Archive".
CMI_REPLACE_WRONG_PRD_AUTO	If set to "ON" the confirmation dialog for the following functionality is disabled: During a Read, if CMI recognizes that a file with a different UUID has been received from the PDM system instead of the file UUID named in the parent CATProduct, then the new file is attached in place of the old.
CMI_ENABLE_DEACTIVATED_CHECK	If set to "ON" the active window is searched for deactivated products, if deactivated are found, the update operation is not allowed.
CMI_ENABLE_PARTINFO_FOR_MODEL	If set to "ON" the more info button will provide information about the Part in Teamcenter (instead of only the data item).
CMI_DISABLE_DT_IN_ARCHIVE	If set to "ON" CMI does not store Design Tables in CMIArchive even if CMI_DESIGN_TABLES is set in Teamcenter.
CMI_USE_DTFORPRODUCT	If set to "ON" CMI also handles Design Tables for CATProducts if CMI_DESIGN_TABLES is set in Teamcenter.
CMI_REP_FORMATS	Sets the Representation formats, which should be handled. Example: "{wrl} {stl}".
CMI_DISABLE_REP_IN_ARCHIVE	If set to "ON" the support for Representations in CMIArchives is disabled.
CMI_REMOVE_UNKNOWN_FILES_IN_XMAP	If set to "ON" unknown files in CMIXMAP will be subject to cache size management.
CMI_XMAP_CACHE_SIZE	Maximum size in MB of cached files in CMIXMAP after closing CATIA.
CMI_RELMAP_CACHE_SIZE	Maximum size in MB of cached cgr files in CMI_RELEASEDCACHEDIR.
CMI_DISABLE_CMISAVELOCALCMD	If set to "ON" the “Save Local” command is not available.
CMI_DISABLE_CMIRESTORELOCALCMD	If set to "ON" the “Restore Local” command is not available.
CMI_USE_91_TOOLBAR	If set to "ON" the 9.1 Toolbar style (only one toolbar) is used.
CMI_GET_BOMTYPE_FROM_TC	Set to "ON" to get the Bom-Type from Teamcenter for new CATIA files.
CMI_ENABLE_CMICATALOGCMD	Set to "ON" to enable the "Manage Catalogs" command in CATIA V5.
CMI_ENABLE_CMICATALOGREADCMD	Set to “ON” to enable the “Read Catalog” command in CATIA V5.
CMI_ENABLE_CMICATALOGUPDCRECMD	Set to “ON” to enable the “Update or Create Catalog” command in CATIA V5.
CMI_ENABLE_CMICATALOGINSERTSCRIPTCMD	Set to "ON" to enable the "Insert CATScript from Teamcenter" command in CATIA V5.

CMI_DELETE_STALE_DESIGNTABLES	If set to "ON" Design Tables in CMIXMAP that are no longer referenced by a CATPart/CATProduct will be deleted.
CMI_USE_FILENAME_WINDOWTITLE	If set to "ON" the title of windows loaded by CMI shows "CMI - <Filename>" instead of the default: "CMI - <PartNumber>".
CMI_ENABLE_CMICHECKINOUTCMD	If set to "ON" enable “Check In/Check Out” command in CATIA V5.
CMI_ENABLE_CMIMODONCREATECMD	Set to "ON" will enable the “Create ModOn” command in CATIA V5.
CMI_ENABLE_CMIALTREPCREATECMD	Set to "ON" will enable “Create AltRep” command in CATIA V5.
CMI_MODON_AUTOUPDATE	Set to "ON" will result in an automatic update after executing the "Create ModOn" command.
CMI_ENABLE_CMICATDUAREAD	Set to "ON" will enable the “CMICatDuaRead” command in CATIA V5.
CMI_CATDUAV5_CONFIG	Full path to the CATDUAV5 configuration file.
CMI_CATDUAV5_RESULTVIEWER	Viewer to show the CATDUA result (Windows optional/unix required). If set to "OFF" no result is shown.
CMI_CATDUAV5_COMMAND	Command to start CATDUAV5 (optional). Default: catstart -run "CATBatchStarter -input @CONFIG@ -output @OUTDIR@".
CMI_ARCHIVE_ALLOW_BROKEN_LINKS	If set to "ON" broken links in the CMIArchive are ignored. Default is "OFF" - broken links are not allowed in a CMIArchive.
CMI_PREVENT_DIFFERENT_VERSIONS	If set to "ON" "To Catia" is checked against different versions of same file in CATIA and in WB.
CMI_DYNAMIC_CATALOG	If set to "ON" CMI stores the Part Master ID in CATCatalogs. “Manage Catalogs” opens always the latest version of the Part, and not a static revision.
CMI_READ_INERTIA	If set to “ON” inertias from CATParts and CATProducts will be sent to Teamcenter, if set to “ONLY_CATPART”, only the inertias of CATParts will be sent to Teamcenter.
CMI_CONFINE_INERTIA_TO_MAINBODIES	If set to “ON” the inertias sent to Teamcenter are confined to those of main part bodies.
CMI_ENABLE_CRE_ANALYSIS_ARCHIVE	If set to "ON" the button “CreateArchive” is enabled, if the top-level-node is a CATAnalysis.
CMI_ANALYSIS_ARCHIVE_OWN_WINDOW	If set to "ON" CMIArchives with CATAnalysis as top-level-node will load into an own window in CATIA.
CMI_NEW_SYNCHRONIZE	Set to "ON" will enable the new SynchronizeCmd and will disable the old Synchronize, update and Create functions.
CMI_SKIP_RO_TEMPLATE	If set to ON, Template CATProducts that were read as Read-only will be skipped during synchronize, i.e. they will not cause a warning.
CMI_GETORIGGEO_DESIGNMODE	If set to "ON" GetOrigGeoCmd: If all files of the selected structures are located in the exchange directory ask to

	load the selected assemblies into DesignMode.
CMI_REQUIRED_LIC_FOR_SYNC	If set to "XXX.prd" CMI checks if the configured CATIA license is allocated, if not, CMI tries to allocate the configured license (shareable) while using CMI Synchronize.
CMI_LICFORSYNC_WARNING	If set to "ON" CMI warns the user if the configured license "CMI_REQUIRED_LIC_FOR_SYNC" could not be allocated.
CMI_ENABLE_EDUFLAG_CHECK	If set to "ON" check for educational flag during Update, Synchronize, Create, SaveAs. If Edu flag is set for a file to be saved, the “Save” action will be declined.
CMI_ENABLE_ACTIVATEDEACTIVATE	If set to "ON" the default value of the user option "Deactivate geometry files not from CMI Workbench" is set to checked. To disable Hide/Show also set CMI_DISABLE_HIDESHOW=ON.
CMI_ENABLE_CHECKFOREMBEDDEDCHANGE	If set to "OFF" do not check if there are embedded changes in the structure (Performance).
CMI_GEOPOS_NOLINK	Set to "OFF" if you use geometry positions but do not link the document to the part when a CATPart is created (default is “ON”).
CMI_DISABLE_SYNC_PROCESS	If set to "ON" the “Synchronize” button is disabled if the current window contains a CATProcess.
CMI_HIDESHOW_ADDTOSESSION	If set to "ON" HideShow/ActivateDeactivate only affects CATIA Windows where the Top-Level Items are available in the current CMI Workbench. This behavior can be overruled by sending a special option during Read to CATIA.
CMI_ENABLE_CREATE_IMPORT_FILE	If set to "ON" the check box "Create Import File" is available in the “Synchronize” dialog.
CMI_ENABLE_EXPORTCMD	If set to "ON" the “Export” command is available in CATIA V5.
CMI_EXPORT_STDATTRIBUTES	To set the CATIA standard attributes Nomenclature, Revision, Description or Definition back to the original value of the initial import set the value of CMI_EXPORT_STDATTRIBUTES to the attributes you want to set back. Only used with the “Export” command.
CMI_REMOVE_CONTEXTMENU	Set to "ON" to disable the contextual CMI menu in the CATIA Product Structure.
CMI_ENABLE_CHECKINOUTDLG_REVISEDATASET	Set to "ON" to enable the “Revise” button in the “CheckIn/CheckOut” dialog.
CMI_IGNORE_MODELINSTNAME_AT_READ	Set to "ON" to ignore Teamcenter Instance name of NonBom Geometries during Read.
CMI_EXPORT_CUSTOMIZE_NAMING	Set to “ON” to customize the naming schema for the export.
CMI_ENABLE_CGM	Set to "ON" to enable the Update, Synchronize, Save As and Save for Doc commands for CGM documents in CATIA.
CMI_STORE_ANALYSIS	Set to "ON" to store CATAnalysis Dataset in Teamcenter.
CMI_ARCHIVE_BOM_CHILDREN	Set to "ON" to allow BOM structures under a CMIArchive.
CMI_ENABLE_CMIBROWSEINTCCMD	If set to "ON" the Browse in TC Command is available in CATIA V5.
CMI_ENABLE_CMIREVISECMD	If set to "ON" the Revise Command is available in CATIA V5.
CMI_CHECKSAVED2	If set to "ON" use different method to detect modified

	files. Caution: If used CMI tries much more updates.
CMI_DISABLE_COMPARECMD	Set to "ON" to disable the compare version functionality.
CMI_HIDE_COMPARECMD	Set to "ON" to hide corresponding CMI command in toolbar and menu without deactivating the command. It is possible to use the hidden commands via macro.
CMI_ENABLE_CUSTOMIZATION	Set to “ON” to enable custom callbacks of the CAA customization.